# COLEGIO DE POSTGRADUADOS

**INSTITUCIÓN DE ENSEÑANZA E INVESTIGACIÓN EN CIENCIAS AGRÍCOLAS**

**CAMPUS MONTECILLO**

**POSTGRADO DE SOCIOECONOMÍA, ESTADÍSTICA E INFORMÁTICA**

**ESTADISTICA**

# MEJORA DE LA CALIDAD DE PROCESOS INDUSTRIALES MEDIANTE SIMULACION Y OPTIMIZACION

**BENIGNO ESTRADA DROUAILLET**

**TESIS**

**PRESENTADA COMO REQUISITO PARCIAL**

**PARA OBTENER EL GRADO DE**

## MAESTRO EN CIENCIAS

**MONTECILLO, TEXCOCO, EDO. DE MEXICO.**

**2010**

La presente tesis titulada: **MEJORA DE LA CALIDAD DE LOS PROCESOS INDUSTRIALES MEDIANTE SIMULACION Y OPTIMIZACION,** realizada por el alumno: Benigno Estrada Drouaillet, bajo la dirección del Consejo Particular indicado, ha sido aprobada por el mismo y aceptada como requisito parcial para obtener el grado de:

**MAESTRO EN CIENCIAS**

**SOCIOECONOMIA, ESTADISTICA E INFORMATICA**

**ESTADISTICA**

**CONSEJO PARTICULAR**

**CONSEJERO**

Dra. Martha Elvá Ramírez Guzmán

**ASESOR**

Dr. Francisco Pérez Soto

**ASESOR**

M.C. Alejandro Corona Ambriz

**MONTECILLO, TEXCOCO, EDO. DE MEXICO. ABRIL DE 2010.**

## AGRADECIMIENTOS

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo económico brindado en la realización de mis estudios y por la confianza en nuestra institución y programa de estudios.

Al Colegio de Postgraduados y al programa de Estadística, por haberme brindado la oportunidad de continuar con mi formación profesional.

A todos y a cada uno de los integrantes del consejo particular por su orientación y apoyo en este trabajo.

A la Dra. Martha Elva Ramírez Guzmán mi más sincero agradecimiento y admiración por su apoyo y valioso tiempo dedicado a este trabajo.

A mis maestros, compañeros y a todas aquellas personas que de alguna forma contribuyeron a la culminación de este trabajo.

## DEDICATORIA

**A mi familia.**

**Contenido**

# MEJORA DE LA CALIDAD DE LOS PROCESOS INDUSTRIALES MEDIANTE SIMULACIÓN Y OPTIMIZACIÓN

## RESUMEN

En la industria es común realizar experimentos para optimizar los procesos de producción; sin embargo, se requieren de considerables recursos económicos y tiempo para desarrollar las nuevas tecnologías. La propuesta en este trabajo es emplear las técnicas de simulación Monte Carlo y bootstrap, para disminuir costos y tiempo en la optimización de procesos. Para lograr este propósito se emplearon la optimización aleatoria, la optimización no lineal NLM (Non-Linear Minimization) y la optimización Taguchi. Estas optimizaciones se compararon con el diseño inicial a través de los índices de capacidad del proceso y la función de pérdida de Taguchi. Los índices de capacidad $C_p$, $C_{pk}$, $C_{pm}$, $C_{pmk}$, $C_s$ e índices ISO fueron simulados con Monte Carlo y sus respectivos intervalos de confianza con remuestreo bootstrap. Los resultados indican que solo las optimizaciones aleatoria y NLM logran identificar las regiones óptimas de las variables del proceso que maximizan el cumplimiento de las especificaciones.

**Palabras clave:** Simulación Monte Carlo, bootstrap, índices de capacidad, intervalos de confianza, optimización aleatoria, Taguchi.

# QUALITY IMPROVEMENT OF INDUSTRIAL PROCESSES BY MEANS OF SIMULATION AND OPTIMIZATION

## ABSTRACT

In industry it is common to conduct experiments to optimize production processes, however, require significant financial resources and time to develop new technologies. The proposal in this paper is to use the Monte Carlo simulation techniques and bootstrap, to reduce costs and time in process optimization. To achieve this purpose were used random optimization, nonlinear optimization NLM (Non-Linear Minimization) and Taguchi Optimization. These optimizations were compared with the initial design through process capability indices and Taguchi loss function. Capability indices $C_p$, $C_{pk}$, $C_{pm}$, $C_{pmk}$, $C_s$ and indices ISO were simulated with Monte Carlo and their respective confidence intervals with bootstrap resampling. The results indicate that only random optimization and NLM can identify the optimal regions of the process variables that maximize the compliance with specifications.

**Keywords:** Monte Carlo Simulation, bootstrap, capability indices, confidence intervals, random optimization, Taguchi.

## 1. INTRODUCCIÓN

El riesgo es la constante de incertidumbre que se encuentra presente invariablemente en la resolución de problemas, desarrollo de estrategias, elaboración de pronósticos o toma de decisiones. Una alternativa para minimizar el riesgo, es el diseño de experimentos el cual consiste en evaluar el efecto de varios niveles de un grupo controlado de factores, alternativas o estrategias. Sin embargo, esta opción consume demasiado tiempo y recursos. Una alternativa a esta problemática es la simulación, la cual además de consumir menos recursos, nos permite simular una infinidad de situaciones potenciales.

En este trabajo se propone la utilización de técnicas de simulación que permiten estudiar el riesgo, simulando valores para cada una de las variables contempladas en un modelo estadístico. La idea de esta estrategia es aprovechar la experiencia y el conocimiento que se tiene del proceso para plantear la simulación.

En el análisis de la optimización de procesos industriales, una medida de la habilidad de un proceso para cumplir con las especificaciones de garantía de calidad, que exige el cliente es el índice de capacidad del proceso (ICP). En la literatura clásica se aborda el cálculo de los índices de capacidad haciendo el supuesto de que el proceso se ajusta a una distribución normal. Sin embargo, muchos procesos no tienen dicha distribución y como consecuencia distorsionan el significado del índice (Kotz y Johnson, 2002). En este trabajo se relaja este supuesto, para dar oportunidad a la inclusión de situaciones reales.

Existen diversos trabajos que abordan la no normalidad de los procesos. Por ejemplo, Collins (1995) propone una técnica de remuestreo para la estimación de intervalos de confianza de índices de capacidad. Kotz y Johnson (2002), hacen una revisión de ICP y dentro de esta revisión hacen referencia a Clements (1989) quien sugiere remplazar $6\sigma$ por la longitud entre los percentiles superior e inferior 0.135 de la distribución del proceso. Tang y Than (1999), revisan algunos métodos para manejar el problema de no normalidad en los ICP y los comparan a través de datos simulados que siguen distribuciones Weibull y log-normal. Ding (2004), presenta un método para estimar ICP para un conjunto de datos a

partir de sus primeros cuatro momentos (media, desviación estándar, coeficiente de simetría, y curtosis). Wu y Swain (2001) proponen un método para manejar la no normalidad del proceso, este método se basa en ponderar la varianza y tiene resultados satisfactorios cuando se trata exclusivamente de la distribución log-normal.

En este trabajo se propone un procedimiento desarrollado en R, que considera el análisis y aplicación de dos técnicas novedosas en el ámbito de la estadística, que son la simulación y la optimización, para modelar el riesgo. Se construye la función de pérdida de Taguchi y los ICP basados en la idea propuesta por Clements para evaluar la eficiencia de los procesos. Los resultados demuestran que ambas técnicas son útiles para mejorar la calidad de los procesos, situación que no sucede para la optimización Taguchi.

## 2. OBJETIVOS

### 2.1. Objetivo General

- Optimizar la calidad de los procesos mediante la aplicación de métodos de remuestreo bootstrap y Monte Carlo a través del cálculo y análisis de Índices de Capacidad y sus respectivos intervalos de confianza.

### 2.2. Objetivos particulares

- Revisar la teoría de los métodos de remuestreo como una alternativa para analizar y comparar los índices de capacidad.
- Simular los procesos empleando simulación Monte Carlo.
- Construir intervalos de confianza bootstrap para los índices de capacidad considerando escenarios en los que el proceso no cumple el supuesto de normalidad.
- Aplicar las técnicas: Optimización Aleatoria, Non-Linear Minimization (NLM) y Optimización Taguchi, para optimizar los procesos.
- Mostrar las ventajas y desventajas de la metodología propuesta sobre el enfoque clásico, mediante simulación estadística.

# 3. ANÁLISIS DE RIESGO

## 3.1 Introducción

Slovic *et al.* (2004), indican que el concepto de riesgo proviene del desconocimiento de lo que ocurrirá en el futuro en respuesta a una acción tomada el día de hoy. El riesgo implica que al tomar una decisión debemos considerar más de un resultado posible.

El objetivo del análisis de riesgo es ayudar en la toma de decisiones para elegir un curso de acción, proporcionando un mejor entendimiento de los posibles resultados que podrían ocurrir (Clemen y Reilly, 1999).

En este sentido, cada acción tiene un riesgo, desde cruzar la calle hasta invertir en acciones de una empresa. Sin embargo, el término usualmente se reserva para situaciones en las que el alcance de los posibles resultados para la acción tomada es de alguna manera trascendente, como el realizar una inversión, lo cual involucra un riesgo importante.

## 3.2 Características del riesgo

El riesgo se deriva de nuestra falta de capacidad de observar lo que sucederá en el futuro, e indica un grado de incertidumbre. Según Clemen y Winkler (1999), el riesgo se divide en objetivo y subjetivo, al primero se le puede asociar una probabilidad de ocurrencia conocida, mientras que al segundo no. Otro de los aspectos que clasifican al riesgo en importante o no es el recurso involucrado, tales como los recursos económicos o de salud.

## 3.3 Evaluación y cuantificación del riesgo

Darse cuenta de que se tiene una situación de riesgo es el primer paso para evaluar y cuantificar el riesgo. ¿Cómo se cuantifica el riesgo que se ha identificado para una situación de incertidumbre? Cuantificar el riesgo significa determinar todos los posibles valores que una variable de riesgo podría tomar y determinar la probabilidad relativa de cada valor. Así

por ejemplo los valores que puede tomar una moneda lanzada al aire son dos, águila y sol y si se trata de una moneda legal, podemos pronosticar que la probabilidad de ocurrencia de cada uno de estos valores es de 0.5. Alternativamente, se podría calcular matemáticamente este resultado haciendo uso de conocimientos básicos de probabilidad y estadística.

En la mayoría de los casos, no se puede efectuar un experimento para calcular su riesgo como se hizo en el caso del lanzamiento de la moneda. Así por ejemplo nos podríamos preguntar ¿Cómo se calcularía la distribución de la presencia de plaga cuarentenaria introducida por productos de origen vegetal de importación? En efecto, se podría calcular la distribución estadística de la presencia de dicha plaga, mediante el mapeo de los datos de los últimos años, y de esta manera la incertidumbre disminuiría. Lo anterior, nos sugiere que la única manera de estimar el riesgo es mediante el apoyo de datos confiables al alcance.

En la mayoría de los casos la cuantificación del riesgo, es imposible por la dificultad que significa realizar repetidamente un experimento para cuantificar la probabilidad de ocurrencia de los posibles valores de un fenómeno. Ante tal situación, una opción es la utilización de distribuciones de probabilidad, como se describe a continuación.

### 3.4 Descripción del riesgo con una distribución de probabilidad

Si se ha cuantificado el riesgo, es decir, si se han determinado los resultados y las probabilidades de ocurrencia, el riesgo se puede resumir mediante una distribución de probabilidad. Una distribución de probabilidad es una función que estima el riesgo de ocurrencia de un evento. Algunas distribuciones estadísticas más conocidas son: normal, uniforme, triangular, gama y la weibull.

Todas las distribuciones se caracterizan por constantes desconocidas denominados parámetros, las cuales definen la forma de la distribución. En la distribución normal, por ejemplo, la media define el valor alrededor del cual está centrada la curva y la desviación estándar define el rango de valores en torno a la media. Por lo tanto, al contar con

información sobre el comportamiento del riesgo probable de una variable, se puede modelar este comportamiento con una distribución. Así, si se sabe que el comportamiento de un riesgo es igual para todos y cada uno de los puntos que toma una variable, entonces este comportamiento se modela con la distribución uniforme.

# 4. INDICES DE CAPACIDAD

## 4.1 Introducción

Poder identificar qué porcentaje de la producción está fuera de especificaciones, es fundamental para que una empresa sea competitiva en un mercado exigente como el actual, y una manera de expresar esta situación es a través de los índices de capacidad de proceso (Guevara y Vargas, 2006).

Los índices de capacidad son estimaciones numéricas de la capacidad del proceso, es decir, nos dan una idea de a qué nivel cumple con las especificaciones (Chen *et al.* 2003). Estos estadísticos son muy útiles ya que, además de ser sencillos de calcular, no tienen unidades de medida, por lo que permiten comparar distintos procesos e índices. Básicamente, son la distancia entre los límites de tolerancia o límites de especificación, y la amplitud real o natural del proceso. La amplitud del proceso, habitualmente se calcula como 6 veces la desviación estándar.

Un análisis de capacidad clásico supone que el proceso se encuentra bajo control estadístico y que la distribución del proceso es normal. Sin embargo, con relación al último supuesto, muchos procesos no tienen dicha distribución y como consecuencia distorsionan el significado del índice (Kotz y Johnson, 2002).

Poder realizar análisis de capacidad bajo la situación de no normalidad permite suministrar a las empresas herramientas confiables para medir su nivel de cumplimiento de estándares externos y facilitar la toma de decisiones con el objetivo de mejorar la calidad de los procesos.

## 4.2 Notación e índices de capacidad del proceso

En lo que se refiere a la notación, por conveniencia, denotaremos los límites de especificación superior e inferior por $USL$ y $LSL$, respectivamente. Los posibles valores

de un proceso los denotaremos por la variable $X$ . El valor esperado y la desviación estándar de $X$ las denotaremos por $\mu$ y $\sigma$ , respectivamente.

### 4.2.1 Índices de capacidad

Los índices de capacidad presentan las siguientes características (Collins, 1995):

- Ayudan a enfatizar la necesidad de mejorar el proceso, mediante la reducción de la variabilidad del proceso.
- Facilitan la comparación del desempeño de distintos proveedores o procesos.
- Proporcionan una idea aproximada del porcentaje de artículos que no cumplen con las especificaciones.

#### 4.2.1.1 Índice $C_p$

Juran (1974) introdujo el índice $C_p$ como una razón, esto es:

$$C_p = \frac{USL - LSL}{6\sigma} = \frac{d}{3\sigma} \qquad (1)$$

Este índice, calcula lo que el proceso es capaz de producir dentro de las especificaciones si el proceso está centrado.

#### 4.2.1.2 Índice $C_{pk}$

El índice $C_{pk}$ , introducido por Kane (1986):

$$C_{pk} = \frac{d - |\mu - M|}{3\sigma} \qquad (2)$$

donde $M = (USL + LSL)/2$, calcula lo que el proceso es capaz de producir dentro de las especificaciones si el objetivo del proceso está centrado alrededor de la media de los limites de especificación. El índice $C_{pk}$ es igual a $C_p$ cuando la media del proceso se ubica en el punto medio de las especificaciones. Si el proceso no está centrado, entonces el valor del índice de $C_{pk}$ será menor que el $C_p$, ya que en tal caso $C_p$ sobreestima la capacidad del proceso. Un valor de $C_{pk} < 0$ indica que la media del proceso se sitúa fuera de los limites de especificación. Al índice $C_{pk}$ también se le suele llamar índice de capacidad real.

Al emplear los índices $C_p$ y $C_{pk}$ se asume que el proceso se distribuye normalmente y que el valor objetivo está centrado respecto a los límites de especificación (Pearn *et al.* 2002).

#### 4.2.1.3 Índice $C_{pm}$

El índice $C_{pm}$ se debe a Hsiang y Taguchi (1985), y en muchas ocasiones se le llama índice de Taguchi. Calcula la capacidad del proceso respecto a un objetivo, $T$. Una de las características del índice $C_{pm}$ es que siempre es mayor que cero.

$$C_{pm} = \frac{d}{3\sqrt{\sigma^2 + (\mu - T)^2}} \qquad (3)$$

Donde $T$ es el valor objetivo. Usualmente, $T = M$; si $T \neq M$ entonces se tiene un caso de tolerancias asimétricas.

#### 4.2.1.4 Índice $C_{pmk}$

El índice $C_{pmk}$ es un índice hibrido que se construye usando el numerador del índice $C_{pk}$ y el denominador del índice $C_{pm}$.

$$C_{pmk} = \frac{d - |\mu - M|}{3\sqrt{\sigma^2 + (\mu - T)^2}} \qquad (4)$$

Choi y Owen (1990), introdujeron este índice. El cual acepta una representación muy interesante en la que están involucrados los índices $C_{pm}$, $C_{pk}$ y $C_p$:

$$C_{pmk} = \frac{C_{pm} C_{pk}}{C_p}$$

El índice $C_{pmk}$ calcula la capacidad del proceso respecto a un objetivo, $T$ para un proceso con una media descentrada.

#### 4.2.1.5 Índice $C_s$

Wright (1995), sugirió el índice $C_s$ como un índice de capacidad del proceso sensible a la asimetría, donde $\mu_3 = E[(X - E(X))^3]$ es una medida de asimetría.

$$C_s = \frac{d - |\mu - M|}{3\sqrt{\sigma^2 + (\mu - T)^2 + |\mu_3 / \sigma|}} \qquad (5)$$

Este es uno de los índices que más sanciona a los procesos ya que considera el centrado de la media con respecto a los límites de especificación, el centrado de la media con respecto a un valor objetivo y además tiene la particularidad de considerar al tercer momento central para medir la asimetría del proceso.

### 4.2.2 Índices de capacidad basados en los percentiles de la distribución del proceso

Clements (1989), en un artículo influyente, sugirió que $6\sigma$ fuera reemplazado por la longitud del intervalo entre los percentiles 0.135 superior e inferior de la distribución de $X$

(analogía a $6\sigma$ para una distribución $N(\mu,\sigma^2)$). De esta manera propuso los índices $1^*$ y $2^*$, a los que llamaremos índices ISO.

$$C_{p\_iso} = \frac{2d}{\xi_{1-a} - \xi_a} \qquad (1^*)$$

Donde $\xi_a$ está definido por $P[X \le \xi_a] = a$, tomando $a = 0.00135$, de este modo $\xi_{1-a}$ y $\xi_a$ son los percentiles 0.135 superior e inferior respectivamente de la distribución de $X$. [Para una distribución $N(\mu,\sigma^2)$, $\xi_{1-a} = \mu + 3\sigma$ y $\xi_a = \mu - 3\sigma$.]

La definición correspondiente para $C_{pk\_iso}$ es:

$$C_{pk\_iso} = \frac{d - \left|\xi_{0.5} - M\right|}{0.5(\xi_{1-a} - \xi_a)} \qquad (2^*)$$

La sustitución del valor esperado, $\mu$, por la mediana, es una elección natural, aunque no esencial.

Siguiendo la idea de Clements se propone la construcción de los índices $C_{pm\_iso}$, $C_{pmk\_iso}$ y $C_{s\_iso}$, correspondientes a los índices $C_{pm}, C_{pmk}$ y $C_s$ respectivamente, los cuales se recomienda utilizar cuando nos interesa medir el cumplimiento de las especificaciones, de un proceso asimétrico y el valor objetivo no está centrado en el rango de las tolerancias.

$$C_{pm\_iso} = \frac{d}{3\sqrt{\left[(\xi_{1-a} - \xi_a)/6\right]^2 + (\xi_{0.5} - T)^2}} \qquad (3^*)$$

$$C_{pmk\_iso} = \frac{d - \left|\xi_{0.5} - M\right|}{3\sqrt{\left[(\xi_{1-a} - \xi_a)/6\right]^2 + (\xi_{0.5} - T)^2}} \qquad (4^*)$$

$$C_{s\_iso} = \frac{d - \left|\xi_{0.5} - M\right|}{3\sqrt{\left[\left(\xi_{1-a} - \xi_a\right)/6\right]^2 + \left(\xi_{0.5} - T\right)^2 + \left|6\,\mu_3/\left(\xi_{1-a} - \xi_a\right)\right|}} \qquad (5^*)$$

## 5. SIMULACION

### 5.1 Introducción

La simulación estadística, es una de las áreas de mayor crecimiento que se ha ido constituyendo como una de las herramientas fundamentales en la toma de decisiones. Existen dos aspectos principales que han contribuido a su crecimiento durante la última década; el incremento en la velocidad del computador y la reducción de los costos de las computadoras, y el desarrollo de herramientas de simulación fáciles de usar, las cuales proporcionan resultados estadísticos completos.

Banks (1998), indica que con la simulación básicamente se busca imitar, con la ayuda de un computador, la operación de uno o varios procesos, con la meta de estimar sus características reales. El proceso en cuestión usualmente se denomina sistema y con el propósito de estudiarlo se hacen supuestos sobre su operación. Estos supuestos, que usualmente toman la forma de relaciones matemáticas o lógicas constituyen el modelo, el cual es usado para tratar de entender cómo se comporta el sistema.

En algunos casos las relaciones que componen un modelo son lo suficientemente sencillas como para utilizar métodos matemáticos, tales como el cálculo, el álgebra o la teoría probabilística. Estos métodos matemáticos ayudan a obtener información exacta sobre aspectos de interés, lo cual se conoce como soluciones analíticas. Sin embargo, la mayoría de los sistemas reales son demasiado complejos para ser evaluados analíticamente, por lo tanto, estos modelos deben ser estudiados por medio de la simulación estadística.

Law y Kelton (2004), proponen desde un punto de vista práctico, los siguientes pasos para llevar a cabo una simulación:

- Definir claramente el problema.
- Introducir las variables asociadas con el problema.
- Elaborar la estructura numérica y traducir el problema a un modelo.

- Preparar posibles cursos de acción por probar.
- Correr el experimento.
- Considerar los resultados, para posiblemente modificar la planeación o entrada de datos.
- Decidir el curso de acción a tomar.

## 5.2 Métodos de simulación

El enfoque o método de simulación determinará la técnica que deberá utilizarse. Dicha técnica dependerá, principalmente, del problema a simular, del objetivo de la simulación y de la potencia de la herramienta informática a emplear. En este trabajo se emplearán dos métodos de simulación, bootstrap y simulación Monte Carlo.

### 5.2.1 Bootstrap

#### 5.2.1.1 Introducción

En muchos procedimientos estadísticos, es necesario conocer determinadas características de la distribución muestral de los estadísticos o los estimadores empleados, con el fin de realizar inferencia estadística. Así, por ejemplo, para la construcción de intervalos de confianza se requiere conocer la varianza del estimador puntual mientras que en el caso del contraste de hipótesis se necesitan los percentiles de la distribución muestral del estadístico de prueba.

Los métodos de remuestreo reemplazan las derivaciones teóricas del enfoque clásico por la evaluación de los estadísticos en remuestras obtenidas a partir de los datos originales, y mediante estos valores se obtienen estimadores de la distribución muestral del estadístico o estimador. El método de remuestreo bootstrap de Efron es uno de los más populares en la literatura estadística junto con el jackknife de Quenouville y Tukey (Alonso, 2002).

Una característica básica del método bootstrap es el principio plug-in, que puede interpretarse como la sustitución de la distribución subyacente $F$ por un estimador $\hat{F}$ de ésta. La selección más frecuente ha sido la función de distribución empírica $\hat{F}(x)$ que está dada por:

$$\hat{F}(x) = F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I_{\{X_i \le x\}}, \qquad (6)$$

donde $I_{\{X_i \le x\}} = 1$ si $X_i \le x$ y $0$ en otro caso (Efron, 1979a; Efron y Tibshirani, 1993).

### 5.2.1.2 Algoritmo del Bootstrap no paramétrico

Solanas y Sierra (1992), indican que en muchas ocasiones no es realista suponer conocida la familia paramétrica a la que pertenece la función de distribución $F$ de la que proceden los datos. En estos casos es necesario estimar $F$ sin hacer hipótesis paramétricas previas. Dada una muestra $X_1,..., X_n$, un estimador no paramétrico sencillo de $F$ es la función de distribución empírica descrita en (6), $\hat{F}(x) = F_n(x)$. Sabemos que $F_n$ es una distribución discreta que asigna probabilidad $1/n$ a cada valor $X_i$. La función de distribución empírica es un estimador consistente de $F(x)$ en el sentido de que, según el teorema de Glivenko-Cantelli, si $n \to \infty$ entonces

$$\sup_x \left| F_n(x) - F(x) \right| \to 0. \qquad (7)$$

Según Efron (1979a), para obtener una muestra $X_1^*,..., X_n^*$, de la distribución $F_n$, hay que llevar a cabo $n$ extracciones con reemplazamiento de la muestra original $X_1,..., X_n$. Por tanto, las observaciones de $X_1^*,..., X_n^*$ y las de $X_1,..., X_n$ coinciden, pero en la muestra artificial puede haber valores de $X_i$ repetidos. Así pues, los datos originales $X_1,..., X_n$ se utilizan una y otra vez para obtener las muestras artificiales. Este hecho justifica el término

remuestreo cuando nos referimos a este procedimiento y a otros similares. El esquema a seguir para aplicar el método bootstrap es el siguiente:

- Se estima $F$ mediante $F_n$. (Principio de sustitución o plug-in.)
- Se obtienen $B$ muestras $X_1^{*b},..., X_n^{*b}$ $(b = 1,..., B)$ procedentes de la distribución $F_n$, seleccionando con reemplazamiento entre los datos originales $X_1,..., X_n$.
- Se calculan los $\hat{\theta}_b$ que en nuestro caso serian por ejemplo, los índices de capacidad $(b = 1,..., B)$ para cada una de las muestras simuladas.
- Después se ordenan $\hat{\theta}_1,..., \hat{\theta}_B$. Al ordenar los datos y sabiendo que cada uno de ellos tiene un peso de $1/n$ entonces podemos calcular los cuantiles que sean de nuestro interés o cualquier función que dependa de $\hat{\theta}_1,..., \hat{\theta}_B$.

### 5.2.1.3 Algoritmo del Bootstrap paramétrico

En el caso del bootstrap paramétrico conocemos la forma funcional de la distribución a la que pertenecen los datos, digamos $f(x,\theta)$. El problema es que, aunque conocemos la familia paramétrica de la que proceden las muestras, desconocemos el valor del parámetro $\theta$. La solución es estimar $\theta$ mediante $\hat{\theta}$, es decir, el estimador que se obtiene aplicando el principio de sustitución. Así, el esquema propuesto por Efron (1979b) es el siguiente:

- Se estima $\theta$ mediante $\hat{\theta}$. (Principio de sustitución o plug-in.)
- Se simulan $B$ muestras de tamaño $n$ procedentes de la distribución $f(x,\hat{\theta})$.
- Se calcula $\hat{\theta}_b$ $(b = 1,..., B)$ para cada una de las muestras simuladas.
- Finalmente se calcula la media, varianza o cualquier función de $\hat{\theta}_1,..., \hat{\theta}_B$.

La diferencia entre el bootstrap paramétrico y el no paramétrico estriba únicamente en la forma de estimar la distribución de la que proceden los datos. El resto del esquema es idéntico.

La selección de $B$ no es un problema sencillo, Efron y Tibshirani (1993) sugieren utilizar $B$ entre 50 y 200 para estimadores de momentos, y al menos 1’000 remuestras para estimadores de distribuciones o cuantiles. En este trabajo se hicieron pruebas con 1’000, 10’000, 100’000 y 1’000’000 de remuestras. Sin embargo, solo en los casos de 100’000 y 1’000’000 de remuestras se alcanzó el nivel de confianza propuesto (95%). De esta manera se eligió $B$ igual a 100’000 para realizar el estudio.

### 5.2.2 Simulación Monte Carlo

#### 5.2.2.1 Introducción

El método fue llamado así por el principado de Mónaco por ser “la capital del juego de azar”, lugar en el que se puede encontrar fácilmente una ruleta que sirve como un generador simple de números aleatorios. El nombre y el desarrollo sistemático de los métodos de Monte Carlo datan aproximadamente de 1944 con el desarrollo de la computadora.

El uso real de los métodos de Monte Carlo como una herramienta de investigación, proviene del trabajo de la bomba atómica durante la Segunda Guerra Mundial. Este trabajo involucraba la simulación directa de problemas probabilísticos de hidrodinámica concernientes a la difusión de neutrones aleatorios en material de fusión. Los orígenes de esta técnica están ligados al trabajo desarrollado por Stan Ulam y John Von Neumann a finales de los 40 en el laboratorio de Los Álamos, cuando investigaban el movimiento aleatorio de los neutrones (Aspray, 1993).

En años posteriores, la simulación Monte Carlo se ha venido aplicando a una infinidad de ámbitos como alternativa a los modelos matemáticos exactos, o incluso como único medio para estimar soluciones para problemas complejos. Así, en la actualidad es posible encontrar modelos que hacen uso de simulación Monte Carlo en las áreas como: informática, empresarial, económica, industrial e incluso social (Judge, 1999). En otras

palabras, la simulación de Monte Carlo está presente en todos aquellos ámbitos en los que el comportamiento aleatorio o probabilístico desempeña un papel fundamental.

Bajo el nombre de Método Monte Carlo o Simulación Monte Carlo se agrupan una serie de procedimientos que analizan distribuciones de variables aleatorias usando simulación de números aleatorios. El Método Monte Carlo da solución a una gran variedad de problemas matemáticos haciendo experimentos con muestreos estadísticos en una computadora.

#### 5.2.2.2 Algoritmo de la simulación Monte Carlo

En principio debemos identificar que el sistema que se intenta simular tiene al menos una variable aleatoria y que se conoce la función de distribución de probabilidad, incluyendo los parámetros, de cada una de las variables aleatorias. Enseguida se explican estas ideas más ampliamente.

La clave de la simulación Monte Carlo consiste en crear un modelo matemático del sistema, proceso o actividad que se quiere analizar, identificando aquellas variables de entrada del modelo cuyo comportamiento aleatorio determina el comportamiento global del sistema. Una vez identificadas dichas variables aleatorias de entrada, se lleva a cabo un experimento que consisten en (1) generar con ayuda del ordenador muestras aleatorias para dichas variables de entrada, y (2) analizar el comportamiento del sistema ante los valores generados. Tras repetir *n* veces este experimento, se dispondrá de *n* observaciones sobre el comportamiento del sistema, lo cual será de utilidad para entender el funcionamiento del mismo, el análisis será tanto más preciso cuanto mayor sea el número *n* de experimentos que se lleven a cabo.

Un esquema típico para realizar simulación Monte Carlo es el siguiente (Judge, 1999):

- Diseñar el modelo lógico de decisión.
- Especificar distribuciones de probabilidad para las variables aleatorias relevantes.
- Incluir posibles dependencias o interacciones entre variables.

- Muestrear valores de las variables aleatorias.
- Calcular el resultado del modelo según los valores del muestreo y registrar el resultado.
- Repetir el proceso hasta tener una muestra estadísticamente representativa.
- Obtener la distribución de frecuencias del resultado de las iteraciones.
- Calcular la media, varianza o cualquier función que sea de nuestro interés de la distribución de frecuencias del resultado de las iteraciones.
- Analizar los resultados.

# 6. METODOS DE OPTIMIZACION

## 6.1 Introducción

La optimización consiste en minimizar la distancia a $f(x)$, donde:

$$f(x),\ x \in \Omega \subseteq \Re^n,$$

En donde, $f(x)$ es la función objetivo, $x = (x_1, ..., x_n)$ es un vector que representa a las variables de decisión y $\Omega$ es el conjunto de decisiones factibles, esto es, las decisiones a las que se restringe la posible solución del problema.

Un problema de optimización trata entonces de tomar una decisión adecuada para maximizar por ejemplo ganancias, velocidad o eficiencia, entre otros, o minimizar costos, tiempo, riesgo o error.

## 6.2 Métodos Taguchi

### 6.2.1 Introducción

Para Taguchi *et al.* (1989), es posible incorporar la calidad en los productos desde su diseño, sin aumentar su costo; los problemas deben eliminarse en el laboratorio de diseño, no en la fábrica o en el campo. Según esta perspectiva, es necesario diseñar productos robustos que toleren variaciones en el proceso de producción, durante el servicio de mantenimiento, transporte y también al mal uso de los consumidores.

Montgomery (1991) menciona que esta metodología sirve para diseñar productos y procesos robustos a las condiciones ambientales o a la variación en sus componentes; y minimiza la variación alrededor de un valor objetivo.

La ingeniería de la calidad de Taguchi combina métodos estadísticos y de ingeniería para optimizar los procesos de diseño y fabricación de modo que aumente la calidad y se reduzcan los costos de los productos. El diseño de experimentos juega un papel esencial en el enfoque de Taguchi, pues ayuda a identificar los factores que intervienen en la generación de problemas de calidad o, alternativamente, los factores que contribuyen a lograr resultados positivos.

### 6.2.2 La función de pérdida y el concepto de calidad

Sullivan (1987), describe como Genichi Taguchi realizó un gran esfuerzo para llevar a un terreno práctico el diseño experimental. Introdujo, además, conceptos revolucionarios que afectaron la forma de medir la calidad y el costo. Para Taguchi, la calidad, antes que el cumplimiento de las especificaciones, debe satisfacer las expectativas del cliente. Situación que mide a través de la "función de pérdida", que establece la pérdida que se genera como consecuencia de la mala calidad. Un producto de calidad es aquél que cumple con las expectativas de rendimiento, cada vez que el cliente lo utiliza, sin fallas y en cualquier condición o circunstancia. Los productos que no cumplen con dichas expectativas causan pérdidas, tanto para los clientes como para los fabricantes.

La función de pérdida de Taguchi, vale cero cuando el desvío con respecto al parámetro objetivo es nulo y se incrementa cuadráticamente cuando los valores de los productos fabricados se acercan a los límites de tolerancia. En otras palabras, los productos cercanos a los límites de tolerancia son productos casi defectuosos y por lo tanto, Taguchi sugiere que los gerentes deberían trabajar para reducir la variabilidad de sus procesos de producción. La función de pérdida se define como:

$$L(y) = k(y - T)^2. \qquad (8)$$

donde:

- $L(y)$ indica la pérdida (en unidades monetarias);

- $k$ es una constante que indica la perdida en unidades monetarias a causa de que la dimensión de calidad de interés se aleje del valor objetivo;
- $T$ es un valor objetivo que la dimensión de interés debe tener; y
- $y$ es el valor de la dimensión de interés de un producto en particular.

### 6.2.3 Métodos recomendados por Taguchi

Taguchi recomienda métodos que se apartan parcialmente de los usados en el diseño de experimentos clásico; la terminología que utiliza también es algo distinta. En primer lugar, Taguchi divide los factores de un experimento en factores controlables y factores incontrolables, o ruido (Roy, 1990).

Según la metodología de diseño de los parámetros, Taguchi recomienda seleccionar dos diseños experimentales, uno para los factores controlables y otro para el ruido. En general, estos diseños son del tipo ortogonal.

Roy (1990), indica que para el análisis de datos, Taguchi recomienda evaluar en el arreglo de los factores controlables la variación de los resultados con una razón señal-ruido apropiada. Se consideran óptimos los niveles de los factores que maximicen una razón señal-ruido adecuada. Estas razones se dividen en nominal es mejor, si el objetivo del experimento es reducir la variabilidad alrededor de un valor objetivo, mayor es mejor si se persigue maximizar una respuesta, y menor es mejor si se desea producir el menor efecto posible. Las diferentes razones señal-ruido son:

- Menor es mejor:

$$SN_S = -10 \log\left(\frac{1}{r}\sum_{i=1}^{r} y_i^2\right) \qquad (9)$$

donde $r$ es el número de repeticiones en una corrida.

- Nominal es mejor:

$$SN_N = 10 \log \left( \bar{y}^2 / S^2 \right) \qquad (10)$$

- Mayor es mejor:

$$SN_L = -10 \log \left( \frac{1}{r} \sum_{i=1}^{r} \frac{1}{y_i^2} \right) \qquad (11)$$

El objetivo principal de la ingeniería es alcanzar mejoras de rendimiento sostenibles ante cualquier condición. Esto es lo que se llama robustez. Quizás uno de los mayores desafíos para Taguchi fue el cómo medir la robustez ya que, sólo si logramos hacerlo, podremos desarrollar tecnologías "a prueba de ruido". Taguchi mide la robustez con la razón señal-ruido. Mientras más robusta es una tecnología, más fuerte es la señal que emite contra cualquier ruido externo que trate de inhibir la fuerza de la señal.

### 6.3 Optimización aleatoria

#### 6.3.1 Introducción

La optimización por búsqueda aleatoria tiene sus orígenes con los trabajos pioneros de Robbins y Monro (1951), quienes sugirieron una técnica iterativa para encontrar las raíces de una función de regresión. Posteriormente, Kiefer y Wolfowitz (1952) sugieren un método para encontrar el valor óptimo de una función.

La efectividad de la optimización aleatoria aumenta conforme se incrementan el número de puntos de búsqueda en la región de interés; es decir, se requiere de un exhaustivo trabajo computacional para aumentar la probabilidad de encontrar el óptimo global.

### 6.3.2 Algoritmo basado en la optimización aleatoria

Para implementar este algoritmo es necesario conocer la distribución de cada una de las variables del modelo a optimizar. El algoritmo de Optimización Aleatoria es el siguiente:

- Se utiliza simulación Monte Carlo para generar vectores de valores aleatorios de cada una de las variables del modelo ($X_1, X_2, \dots, X_n$).
- Los valores obtenidos se introducen en el modelo para estimar la respuesta del proceso ($Y$).
- Se identifican los valores estimados que estén a una distancia $\delta$ del valor objetivo ($T$).
- Se delimita la región optima, esto es para cada variable $X$ se define el intervalo que produce valores de $Y$ cercanos a $T$.
- Finalmente, se encuentra el vector del centro de la región óptima ($X_1^*, X_2^*, \dots, X_n^*$).

### 6.4 Optimización empleando la función NLM (Non-Linear Minimization) de R

Esta función lleva a cabo la optimización de una función $f$ cualquiera, utilizando el algoritmo de Newton. Este algoritmo también es conocido como el método de Newton-Raphson o el método de Newton-Fourier. Este algoritmo localiza el máximo o mínimo de $f(x) = 0$, mediante las raíces de la primera derivada a través de un proceso iterativo (Tjalling, 1995):

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \qquad (12)$$

De esta manera, a partir de un valor inicial, $x_i$, se puede encontrar el siguiente valor, $x_{i+1}$, utilizando la ecuación (12). Este proceso se repite hasta que $\left|x_{i+1} - x_i\right| / \left|x_{i+1}\right| \leq \delta$, para un $\delta > 0$.

## 7. ANALISIS BIBLIOGRAFICO

Al optimizar con las rutinas mencionadas anteriormente se han encontrado algunas desventajas como son:

- Debido al diseño del algoritmo NLM (Ecuación 12), se puede observar la incapacidad que tiene NLM para salir de óptimos locales, situación que es solventada por el algoritmo de optimización aleatoria, que utiliza simulación Monte Carlo para generar vectores que contiene valores aleatorios de cada una de las variables del modelo. Por lo tanto, uno podría esperar que el segundo método de optimización fuera mejor que el primero.

- Por su parte Taguchi necesita de al menos 3 niveles en cada factor para tener una idea más precisa del comportamiento de la curva SN y así maximizarla. Situación que se limitaría cuando solamente se tienen 2 niveles por cada factor de interés.

- El algoritmo aleatorio, requiere de un gran trabajo computacional para optimizar la función de interés, ya que puede suceder el caso de que en pocas iteraciones ya se haya encontrado el máximo deseado, pero por la forma en que trabaja esta rutina seguirá buscando hasta realizar las iteraciones que se le pidieron. Sin embargo, gracias a este trabajo computacional se minimiza el riesgo de elegir un máximo (mínimo) local. Esta situación resulta ventajosa cuando se requiere alcanzar un valor meta, ya que la rutina se puede detener cuando se alcanza una medida de eficiencia, como la función de pérdida de Taguchi.

- En la actualidad se cuenta con software comercial como el @RISK, que emplea simulación Monte Carlo para modelar el riesgo. Algunas desventajas que tiene este software es que en principio, es caro. Otro punto es que no cuentan con ventanas para programar estadísticas de interés como intervalos de confianza bootstrap para índices de capacidad, medidas de eficiencia como la función de pérdida de Taguchi, la función de distribución empírica de los datos y de nuevas estadísticas de interés

como los índices de capacidad: $C_{p\_iso}$ , $C_{pk\_iso}$ , $C_{pm\_iso}$ , $C_{pmk\_iso}$ y $C_{s\_iso}$ . Esto tiene relevancia cuando se desea conocer el comportamiento de los índices de capacidad antes mencionados, bajo diferentes tipos de distribución de la variable de respuesta.

Por lo anterior, a continuación se propone una metodología para mejorar la calidad de un proceso mediante el empleo de la optimización aleatoria, simulación Monte Carlo y Bootstrap. Estrategia que minimiza los costos de operación y tiempo de experimentación.

## 8. METODOLOGIA PROPUESTA

Para el análisis de riesgo, en este trabajo se emplearán métodos cuantitativos que buscan determinar los posibles resultados en una situación de toma de decisiones, en donde las variables de los escenarios siguen una distribución de probabilidad. En general, las técnicas empleadas en este trabajo para el análisis de riesgo abarcan seis etapas programadas en R:

1. Identificación del modelo mediante la teoría clásica
2. Modelación de la incertidumbre
3. Simulación Monte Carlo
4. Construcción de índices de capacidad y de sus respectivos intervalos de confianza
5. Análisis del modelo simulado
6. Optimización del modelo simulado

La metodología que se propone en este trabajo proporciona una herramienta poderosa y flexible que permite construir un modelo óptimo y analizar el riesgo asociado. Los resultados obtenidos pueden ser usados para tomar una decisión y elegir un curso de acción. Se presentara una metodología análoga a la empleada en los límites $6$ sigma, que se basa en los percentiles $0.135$ y $99.885$ de la distribución del proceso para estimar su amplitud natural.

Para dar solución a cada una de las etapas que se plantean en esta metodología se hace uso del software R para programar rutinas que facilitan la toma de las decisiones. A continuación de describe cada una de estas etapas.

### 8.1 Identificación del modelo

Se trata de identificar el mejor modelo que represente la variación de la característica de calidad del producto o servicio. Una vez que se han obtenido algunos valores del proceso, el primer paso es ajustar, con el apoyo de R, un modelo estadístico que represente la variación del proceso en función de posibles variables explicativas del proceso.

### 8.2 Modelación de la incertidumbre

El siguiente paso, consiste en determinar una distribución apropiada para cada una de las variables del proceso o en su defecto para las variables de riesgo dentro del proceso. Para este paso es muy importante la experiencia y el conocimiento de expertos sobre el comportamiento del proceso. Este punto es fundamental, porque al dejar libres los supuestos de las funciones de probabilidad, se tiene la posibilidad de dar solución a una gama muy amplia de problemas de toma de decisiones.

### 8.3 Simulación Monte Carlo

Una vez realizadas las elecciones de las distribuciones es momento de realizar la simulación Monte Carlo del proceso bajo estudio para analizar el mayor número de escenarios posibles.

Se debe notar que al asignar libremente las distribuciones a las variables del proceso, es difícil pensar que se tiene un escenario en el cual el proceso se ajuste a una distribución normal, como se supondría de manera clásica. Inclusive es difícil encontrar resultados analíticos que indiquen la distribución que sigue un proceso en el cual intervienen variables que no necesariamente siguen una misma familia de distribuciones.

La observación anterior no debe desalentarnos, ya que es la motivación de este trabajo, a partir de los datos obtenidos en la simulación Monte Carlo se construirán los Índices de Capacidad del Proceso (ICP) basados en la distribución empírica de dicho proceso, también con la ayuda del remuestreo Bootstrap se construirán los respectivos Intervalos de Confianza (IC).

### 8.4 Construcción de índices de capacidad y de sus respectivos intervalos de confianza

Para la construcción de cada uno de los ICP y de sus respectivos IC se emplea la técnica de remuestreo Bootstrap no paramétrico. En principio se obtiene la función de distribución

empírica de cada uno de los ICP y posteriormente se encuentran los cuantiles necesarios para la construcción de los IC.

### 8.5 Análisis del modelo simulado

Una vez llevado a cabo el análisis de riesgo y considerando los resultados obtenidos, se deben elegir los valores óptimos para cada variable. En otras palabras, una vez que se han construido los ICP, se analiza la fracción defectuosa, es decir, el porcentaje de unidades que están fuera de los límites de especificación.

Basados en el valor de cada uno de los ICP, en los IC, en la fracción defectuosa y en la distribución del proceso; se decide si es necesario optimizar el proceso.

### 8.6 Optimización del modelo simulado

La idea de optimizar el proceso es encontrar los valores de cada una de las variables que intervienen en el modelo, tales que al final se tenga un proceso centrado en el valor meta, y que se disminuya la variación entre las unidades producidas, de tal manera que la mayoría de los resultados estén dentro de los limites de especificación.

Para la etapa de optimización del proceso se emplean tres técnicas: optimización aleatoria, optimización Taguchi y NLM (Non-Linear Minimization). Para cada uno de estos modelos de optimización se calculan los ICP, sus respectivos IC y la fracción defectuosa.

En la Figura 1 se muestra un resumen de la metodología propuesta en la que destaca el trabajo conjunto entre la teórica clásica y las nuevas técnicas desarrolladas como lo es la simulación.

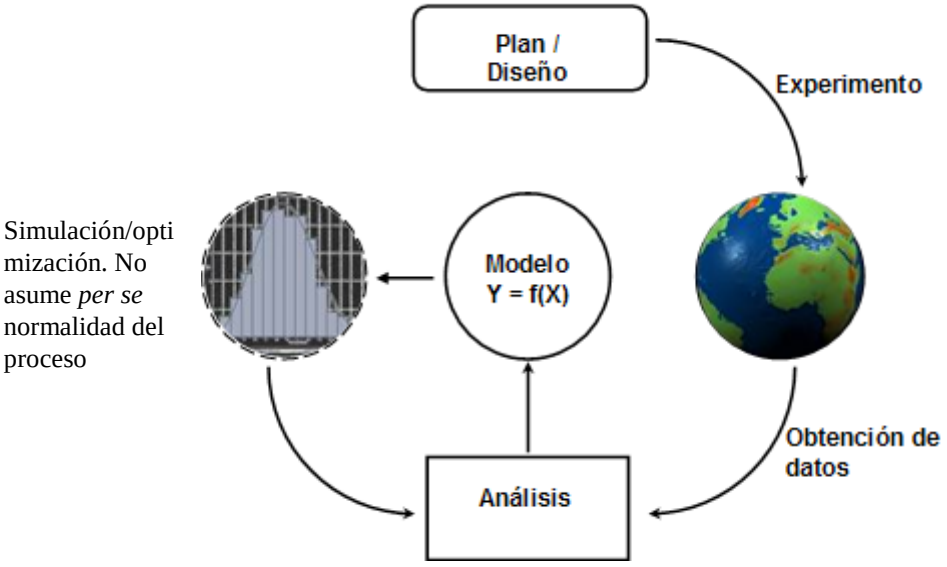


**Figura 1.** Esquema de la metodología propuesta

## 9. APLICACIÓN

Sleeper (2005) presenta la aplicación del software "Crystall Ball" para encontrar los niveles óptimos de tres factores que minimizan la función de pérdida de Taguchi. La intención es fabricar inyectores que inyecten un volumen preciso de combustible, el volumen por ciclo es una de las características de calidad del inyector. La tolerancia para el volumen de combustible es $300 \pm 30$ $mm^3$, especificada por el cliente.

El experimento de Sleeper, consistió en determinar cómo reacciona el volumen a cambios en tres componentes que se cree que son críticos. El objetivo del experimento fue desarrollar un modelo para el volumen como función de estos tres componentes, y entonces usar el modelo para optimizar el sistema. El cliente especifica las siguientes tolerancias para cada uno de los componentes; $\pm 50$ para el factor A, $\pm 0.15$ para el factor B y $\pm 0.03$ para el factor C. El esquema del diseño utilizado se muestra enseguida.

**Cuadro 1.** Factores y niveles del experimento

| Factor | | Niveles experimentales | | |
|---|---|---|---|---|
| | | **Bajo** | **Alto** | **Tolerancias** |
| **A** | **Fuente de carga** | 500 | 900 | ± 50 |
| **B** | **Boquilla de flujo** | 6 | 9 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.3 | 0.6 | ± 0.03 |

El Cuadro 1 muestra los tres factores y dos niveles elegidos para el experimento. Los niveles experimentales son más amplios que las tolerancias para los tres componentes, porque se desea observar cómo es que los factores afectan el volumen en un rango amplio. El Cuadro también enlista los valores de las tolerancias para cada factor.

Como se puede observar, el trabajo de Sleeper se suscribe a la obtención de una optimización puntual, asumiendo normalidad. En contraparte, la propuesta de esta investigación proporciona una región óptima para los niveles de los factores del experimento, la probabilidad de obtener de inyectores dentro de las especificaciones, no asume *per se* normalidad del proceso y además calcula una gama más amplia de índices de capacidad.

## 9.1 Aplicación de la Metodología Propuesta

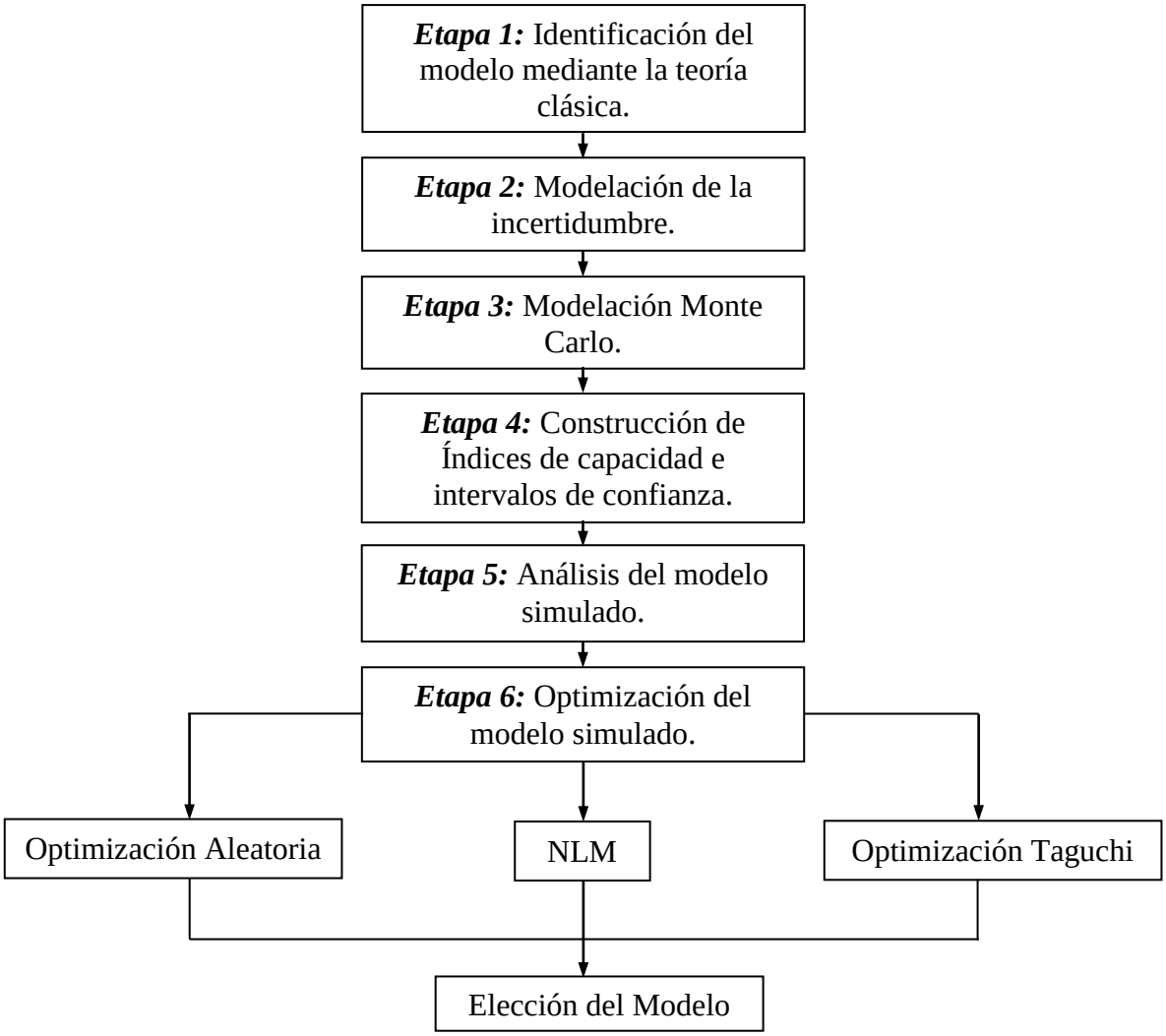


**Figura 2.** Metodología Propuesta

Con el fin de ilustrar las observaciones que se describieron en la sección de análisis de bibliografía y de comparar las metodologías: Clásica, Optimización aleatoria, NLM y Optimización Taguchi, a continuación se presentan los resultados obtenidos en cada una de estas metodologías.

### 9.1.1 Aplicación de la teoría clásica

Se realiza el experimento factorial completo con tres repeticiones, el cual incluye ocho realizaciones que representa todas las combinaciones de tres factores a dos niveles. Así se decide construir un total de 24 inyectores para el experimento, asignando tres inyectores a cada una de las ocho corridas.

**Cuadro 2.** Resultados del experimento

| Realización | Factor/Niveles | | | Volumen Medido | | |
|---|---|---|---|---|---|---|
| | A | B | C | | | |
| **1** | 500 | 6 | 0.3 | 126 | 141 | 122 |
| **2** | 900 | 6 | 0.3 | 183 | 168 | 164 |
| **3** | 500 | 9 | 0.3 | 284 | 283 | 275 |
| **4** | 900 | 9 | 0.3 | 300 | 318 | 310 |
| **5** | 500 | 6 | 0.9 | 249 | 242 | 242 |
| **6** | 900 | 6 | 0.9 | 125 | 128 | 140 |
| **7** | 500 | 9 | 0.9 | 387 | 392 | 391 |
| **8** | 900 | 9 | 0.9 | 284 | 269 | 255 |

Una vez realizado el experimento se ajusta el modelo y se obtienen los resultados del Cuadro 3.

**Cuadro 3.** Análisis del diseño factorial en R

| Coeficiente | Estimado | Error Estándar | t | $\Pr(>\|t\|)$ |
|---|---|---|---|---|
| Intercepto | 240.7500 | 1.7722 | 135.850 | < 2e-16 *** |
| A | -20.4167 | 1.7722 | -11.521 | 3.71e-09 *** |
| B | 71.5833 | 1.7722 | 40.393 | < 2e-16 *** |
| C | 17.9167 | 1.7722 | 10.110 | 2.36e-08 *** |
| AB | -2.5833 | 1.7722 | -1.458 | 0.164 |
| AC | -38.0833 | 1.7722 | -21.490 | 3.15e-13 *** |
| BC | -0.5833 | 1.7722 | -0.329 | 0.746 |
| ABC | 0.7500 | 1.7722 | 0.423 | 0.678 |

Error estándar de los residuales: 8.682 con 16 grados de libertad
R-cuadrada ajustada: 0.9902

En el Cuadro 3 se observa que los coeficientes significativos son; A, B, C y la interacción AC. Después de remover los efectos que no son significativos se vuelve a realizar el análisis en R y se obtienen los resultados que se muestran en el Cuadro 4.

**Cuadro 4.** Análisis del diseño factorial en R usando los coeficientes significativos

| Coeficiente | Estimado | Error Estándar | t | $\Pr(>\|t\|)$ |
|---|---|---|---|---|
| Intercepto | 240.750 | 1.745 | 138.00 | < 2e-16 *** |
| A | -20.417 | 1.745 | -11.70 | 3.95e-10 *** |
| B | 71.583 | 1.745 | 41.03 | < 2e-16 *** |
| C | 17.917 | 1.745 | 10.27 | 3.42e-09 *** |
| AC | -38.083 | 1.745 | -21.83 | 6.45e-15 *** |

Error estándar de los residuales: 8.547 con 19 grados de libertad
R-cuadrada ajustada: 0.9905

Después de remover los efectos no significativos del análisis, la salida de R reporta un valor de $R^2 = 0.99$, es decir el modelo explica el $99\%$ de la variación del volumen de combustible del inyector. R también estima la variación entre inyectores ésta es $s = 8.547$.

Para concluir con esta etapa, se construye el siguiente modelo para representar el volumen de combustible proporcionado por el inyector en función de los tres factores:

$$Y = 240.75 - 20.42\,A + 71.58\,B + 17.92\,C - 38.08\,AC \qquad (13)$$

### 9.1.2 Modelación de la incertidumbre

En este paso se asignan las distribuciones a cada uno de los factores del proceso, para llevar a cabo la simulación de los inyectores en R. Antes de la asignación de las distribuciones se eligen los valores del diseño inicial que en este caso serán los valores que se utilizaron para el primer prototipo. En el Cuadro 5 se muestran los valores del diseño inicial y las tolerancias especificadas por el cliente.

**Cuadro 5.** Diseño inicial y tolerancias

| | Factor | Diseño inicial | |
|---|---|---|---|
| | | **Valor** | **Tolerancia** |
| **A** | **Fuente de carga** | 500 | ± 50 |
| **B** | **Boquilla de flujo** | 6.75 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.6 | ± 0.03 |

Para agilizar la simulación Monte Carlo se codificaran los factores A, B y C de modo que $-1$ indicará el nivel bajo y $1$ el nivel alto de cada factor. La codificación se realizará de la siguiente manera; a cada factor se le restará el valor central de sus límites de tolerancia y esta diferencia se dividirá entre la distancia que existe del centro a los extremos de las tolerancias del factor en cuestión (Cuadro 1). En las expresiones 14, 15 y 16 se muestran las codificaciones para cada uno de los factores.

$$A = \frac{\text{Fuente de carga - 700}}{200} \qquad (14)$$

$$B = \frac{\text{Boquilla de flujo} - 7.5}{1.5} \qquad (15)$$

$$C = \frac{\text{Lanzadera de ascenso} - 0.45}{0.15} \qquad (16)$$

### 9.1.3 Modelación Monte Carlo

Ahora ingresamos el modelo en R para implementar un análisis Monte Carlo. Durante el análisis los valores de A, B, C y la variación entre inyectores son reemplazados por valores generados "aleatoriamente" que se obtienen de sus respectivas distribuciones. Para cada uno de los reemplazos se calcula el volumen proporcionado por el inyector con los parámetros iniciales codificados, es decir, $A = -1,\ B = -0.5$ y $C = 1$.

Puesto que es la primera vez que se trabaja con este proceso y no se tienen mediciones ni observaciones previas respecto al comportamiento de cada uno de los factores, es razonable

pensar que cada uno de los factores se distribuye uniforme dentro de sus tolerancias codificadas. Es decir el factor A se distribuye $U(-1.25,-0.75)$, el factor B se distribuye $U(-0.6,-0.4)$ y el factor C se distribuye $U(0.8,1.2)$. En cuanto a la variación entre inyectores se supone que esta se distribuye normal con media 0 y desviación estándar 8.682.

Después de simular 100’000 inyectores, las unidades resultantes mediante la simulación Monte Carlo, proporcionan los siguientes resultados.

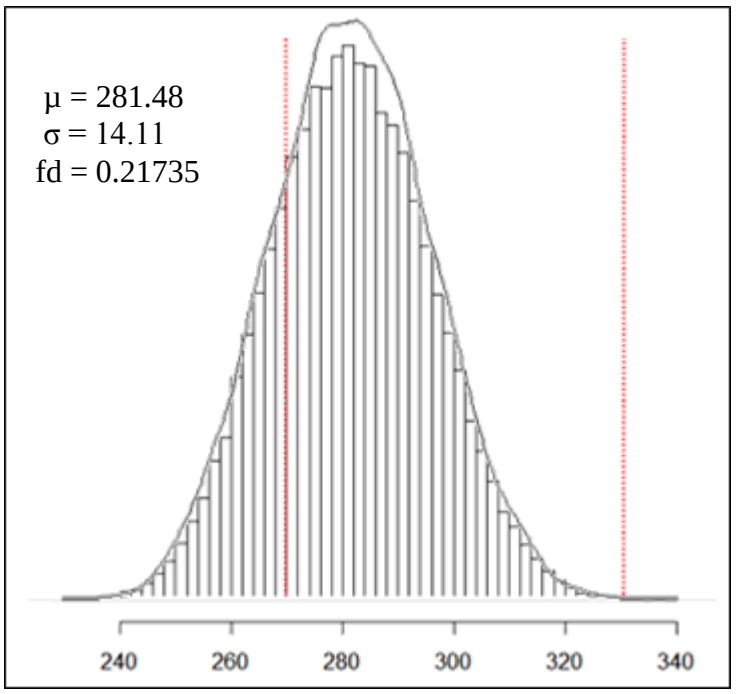


**Figura 3.** Histograma y densidad de la simulación basada en el diseño inicial

En la Figura 3, se observa que el proceso no está centrado en el valor objetivo y en cambio esta sesgado hacia el límite de tolerancia inferior para el volumen, es decir, 270. También se observa que alrededor del 21% de los inyectores no cumplirían con las especificaciones del volumen, y en su mayoría estarían inyectando un volumen inferior al especificado por el cliente. También se observa que la densidad no tiene la apariencia de la típica forma de campana que caracteriza a la distribución normal. R también reporta que la media del proceso es 281.48 y la desviación estándar del volumen inyectado entre estos 100’000 inyectores virtuales es 14.11.

### 9.1.4 Construcción de índices de capacidad

En este punto se construirán y analizarán los ICP y sus respectivos IC obtenidos con la metodología bootstrap no paramétrica que se implementó en R. Los valores obtenidos se muestran en el Cuadro 6. En este Cuadro se nota que el valor de los ICP son muy bajos, lo que indica que el proceso no cumple con las especificaciones.

**Cuadro 6.** ICP y sus respectivos IC

| | **ICP** | **LI** | **LS** |
|---|---|---|---|
| $C_p$ | 0.7007154 | 0.6920878 | 0.7095278 |
| $C_{p\_iso}$ | 0.7445953 | 0.7197919 | 0.7692312 |
| $C_{pk}$ | 0.2650594 | 0.2582275 | 0.2719518 |
| $C_{pk\_iso}$ | 0.2739961 | 0.2614264 | 0.2866653 |
| $C_{pm}$ | 0.4257988 | 0.4216261 | 0.4300273 |
| $C_{pm\_iso}$ | 0.4303535 | 0.4231905 | 0.4374887 |
| $C_{pmk}$ | 0.1610761 | 0.1558912 | 0.1663458 |
| $C_{pmk\_iso}$ | 0.1583781 | 0.1512020 | 0.1656158 |
| $C_s$ | 0.1587325 | 0.1534191 | 0.1641373 |
| $C_{s\_iso}$ | 0.1572481 | 0.1499377 | 0.1646766 |

En el Cuadro 6 se nota que el valor de los ICP son muy bajos lo que indica que el proceso no cumple con las especificaciones, lo cual era de esperarse desde que se observó en la Figura 3 que el proceso producía un gran número de unidades por fuera de las especificaciones, $fd = 0.21$, y más aun que la media del proceso era de 281 cuando el valor objetivo especificado por el cliente es de 300.

### 9.1.5 Análisis del modelo simulado

En esta etapa se ha corrido una rutina para que proporcione los niveles de confianza observados para cada uno de los IC.

**Cuadro 7.** Nivel de confianza observado para cada uno de los IC

| | ICP | ICP_iso |
|---|---|---|
| $C_p$ | 0.9508 | 0.9521 |
| $C_{pk}$ | 0.9455 | 0.9474 |
| $C_{pm}$ | 0.9423 | 0.9461 |
| $C_{pmk}$ | 0.9435 | 0.9453 |
| $C_s$ | 0.9354 | 0.9484 |

En el Cuadro 7 los niveles de confianza observados están alrededor del 95%, este hecho era de esperarse y se debe a que los IC que se construyeron para cada uno de los ICP fueron a un nivel de confianza del 95%. También es claro que el nivel de confianza observado para los ICP es ligeramente menor que para los ICP_iso.

### 9.1.6 Comparación de los métodos de Optimización con el diseño inicial

#### 9.1.6.1 Optimización Aleatoria

En las etapas 3 y 4 se observó que el proceso inicial no es adecuado, ya que el proceso no está centrado en el valor meta y está sesgado a la izquierda, por lo que es conveniente optimizar cada uno de los factores que intervienen en este proceso, para ello se utilizó la optimización aleatoria. El diseño optimizado se muestra en seguida.

**Cuadro 8.** Diseño mejorado y tolerancias

| **Factor** | | **Diseño optimo** | |
|---|---|---|---|
| | | **Valor** | **Tolerancia** |
| **A** | **Fuente de carga** | 749.31 | ± 50 |
| **B** | **Boquilla de flujo** | 9.02 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.28 | ± 0.03 |

En la Figura 4 se observa que el inyector que tenga todos sus componentes dentro de los límites de tolerancia, tendrá una probabilidad de 0.99 de suministrar el volumen de acuerdo a las especificaciones, $300 \pm 30$ mm$^3$.

**a) Simulación Monte Carlo del diseño obtenido con la optimización aleatoria**

Después de que se encontró el diseño óptimo (Cuadro 8), el siguiente paso fue simular 100’000 unidades de este proceso con la técnica Monte Carlo, para corroborar que el proceso justamente se optimizó y observar las mejoras obtenidas en el proceso.

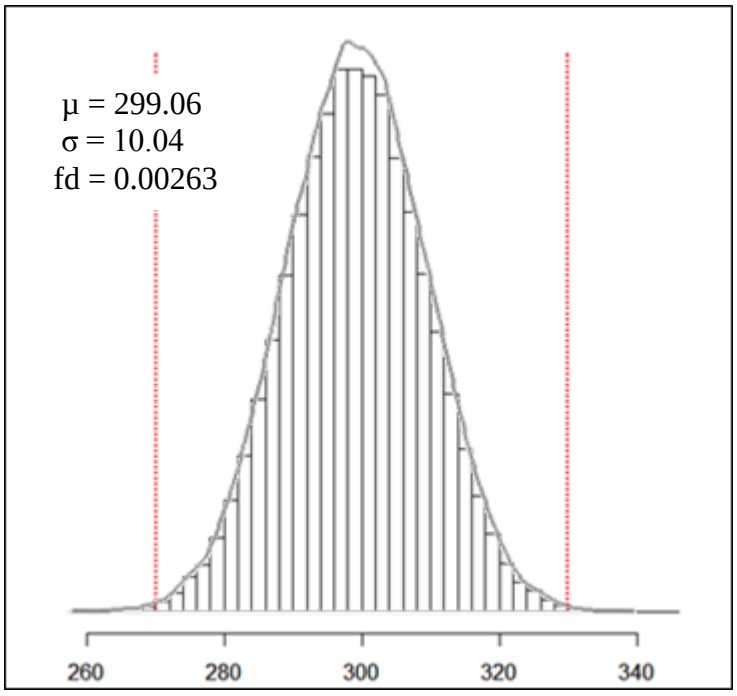


**Figura 4.** Histograma y densidad de la simulación basada en el diseño optimizado

En la Figura 4, se observa a diferencia de la Figura 3 que el proceso ahora está centrado en el valor objetivo. También se observa que de mil inyectores construidos bajo las especificaciones del modelo optimizado, alrededor de 3 no cumplirían con las especificaciones del volumen inyectado.

**b) Cálculo de los ICP y de los IC para el diseño optimizado**

En esta etapa al igual que en la etapa 3 se construyeron y analizaron los ICP y sus respectivos IC, con la diferencia de que en este punto se utilizo el diseño optimizado con la técnica de optimización aleatoria. Los valores obtenidos se muestran en el Cuadro 9.

**Cuadro 9.** ICP y sus respectivos IC para el diseño optimizado

| | ICP | LI | LS |
|---|---|---|---|
| $C_p$ | 0.9988127 | 0.9168882 | 1.0911108 |
| $C_{p\_iso}$ | 1.0128691 | 0.9722956 | 1.0500006 |
| $C_{pk}$ | 0.9660022 | 0.8786586 | 1.0609604 |
| $C_{pk\_iso}$ | 0.9812177 | 0.9402450 | 1.0182702 |
| $C_{pm}$ | 0.9923819 | 0.9111097 | 1.0826729 |
| $C_{pm\_iso}$ | 1.0082354 | 0.9677920 | 1.0450884 |
| $C_{pmk}$ | 0.9599044 | 0.8673906 | 1.0581821 |
| $C_{pmk\_iso}$ | 0.9767348 | 0.9360603 | 1.0141824 |
| $C_s$ | 0.8680468 | 0.7192426 | 0.9983268 |
| $C_{s\_iso}$ | 0.9398769 | 0.8701779 | 0.9956575 |

En el Cuadro 9 se observa que el valor de cada uno de los ICP ha aumentado considerablemente en comparación con los valores que se observaron en el Cuadro 3. Además se nota que el índice Cs es el que más castiga al proceso, lo cual es de esperarse por asumir distribución normal erróneamente.

Es importante mencionar que a pesar de que los índices de capacidad son similares, los intervalos de confianza son diferentes, ya que los intervalos de los índices ISO son más angostos que los clásicos. Esto era de esperarse por la falta del supuesto de normalidad.

**c) Análisis del modelo optimizado**

En esta etapa se corrió una rutina para corroborar el nivel de confianza de cada uno de los intervalos. El Cuadro 10 muestra que en efecto los valores se encuentran cercanos al 95% esperado. Al igual que en el proceso no centrado el nivel de confianza de los ICP es ligeramente menor que los ICP_iso.

**Cuadro 10.** Nivel de confianza observado para cada uno de los ICP

| | ICP | ICP_iso |
|---|---|---|
| $C_p$ | 0.9475 | 0.9505 |
| $C_{pk}$ | 0.9424 | 0.9546 |
| $C_{pm}$ | 0.9447 | 0.9512 |
| $C_{pmk}$ | 0.9438 | 0.9537 |
| $C_s$ | 0.9290 | 0.9523 |

### 9.1.6.2 Optimización No Lineal utilizando la función NLM

El diseño optimizado se muestra en seguida.

**Cuadro 11.** Diseño mejorado y tolerancias

| Factor | | Diseño optimo | |
|---|---|---|---|
| | | **Valor** | **Tolerancia** |
| **A** | **Fuente de carga** | 616.82 | ± 50 |
| **B** | **Boquilla de flujo** | 8.90 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.38 | ± 0.03 |

R reporta que el inyector que tenga todos sus componentes dentro de los límites de tolerancia tendrá una probabilidad de 0.99 de suministrar el volumen de acuerdo a las especificaciones, $300 \pm 30$ mm$^3$.

**a) Simulación Monte Carlo del diseño obtenido con la función NLM**

Después de que se encontró el diseño óptimo el siguiente paso fue simular 100’000 unidades de este proceso con la técnica Monte Carlo para observar las mejoras obtenidas bajo el nuevo diseño (Figura 5).

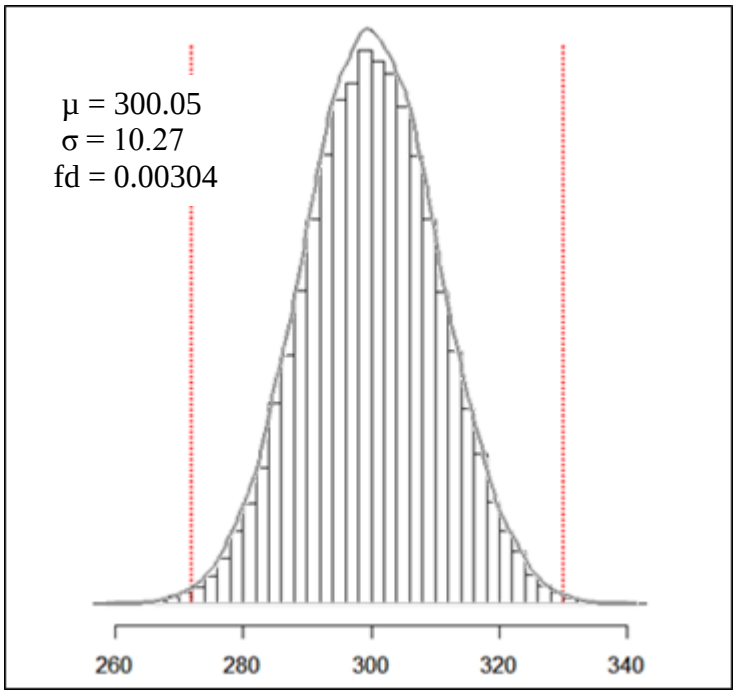


**Figura 5.** Histograma de la simulación basada en el diseño optimizado

Se observa que el proceso está centrado en el valor objetivo, 300. También se observa que de mil inyectores construidos bajo las especificaciones del modelo optimizado con la función NLM, alrededor de 3 no cumplirían con las especificaciones del volumen inyectado.

**b) Cálculo de los ICP y de los IC para el diseño optimizado con NLM**

En el Cuadro 12 se observa que el valor de cada uno de los ICP ha aumentado considerablemente, en comparación con los valores que se observaron en el proceso sin mejorar (Cuadro 3).

**Cuadro 12.** ICP y sus respectivos IC para el diseño optimizado

| | **ICP** | **LI** | **LS** |
|---|---|---|---|
| $C_p$ | 0.9770560 | 0.8937849 | 1.0676730 |
| $C_{p\_iso}$ | 0.9926665 | 0.9606298 | 1.0246933 |
| $C_{pk}$ | 0.9601854 | 0.8763418 | 1.0537046 |
| $C_{pk\_iso}$ | 0.9892384 | 0.9566703 | 1.0218121 |
| $C_{pm}$ | 0.9750871 | 0.8928557 | 1.0660017 |
| $C_{pm\_iso}$ | 0.9925842 | 0.9605396 | 1.0246141 |
| $C_{pmk}$ | 0.9582846 | 0.8717899 | 1.0527823 |
| $C_{pmk\_iso}$ | 0.9891567 | 0.9565103 | 1.0217914 |
| $C_s$ | 0.8661876 | 0.7184381 | 0.9978765 |
| $C_{s\_iso}$ | 0.9506520 | 0.8820685 | 1.0042251 |

También es claro que los índices ISO son más angostos en relación a sus respectivos índices clásicos.

**c) Análisis del modelo optimizado con NLM**

**Cuadro 13.** Nivel de confianza observado para cada uno de los ICP

| | ICP | ICP_iso |
|---|---|---|
| $C_p$ | 0.9510 | 0.9467 |
| $C_{pk}$ | 0.9487 | 0.9485 |
| $C_{pm}$ | 0.9506 | 0.9463 |
| $C_{pmk}$ | 0.9517 | 0.9490 |
| $C_s$ | 0.9484 | 0.9516 |

En el Cuadro 13 los niveles de confianza observados están alrededor del 95%, este hecho era de esperarse y se debe a que los IC que se construyeron para cada uno de los ICP fueron a un nivel de confianza del 95%.

### 9.1.6.3 Optimización Taguchi

El diseño optimizado se muestra en seguida.

**Cuadro 14.** Diseño mejorado y tolerancias

| Factor | | Diseño optimo | |
|---|---|---|---|
| | | **Valor** | **Tolerancia** |
| **A** | **Fuente de carga** | 500 | ± 50 |
| **B** | **Boquilla de flujo** | 9 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.6 | ± 0.03 |

R reporta que todos los inyectores fabricados bajo el diseño especificado en el Cuadro 14 proporcionaran un volumen mayor al requerido.

**a) Simulación Monte Carlo del diseño obtenido con Taguchi**

Después de que se encontró el diseño óptimo el siguiente paso fue simular 100’000 unidades de este proceso con la técnica Monte Carlo para observar los cambios obtenidos bajo el nuevo diseño (Figura 6).

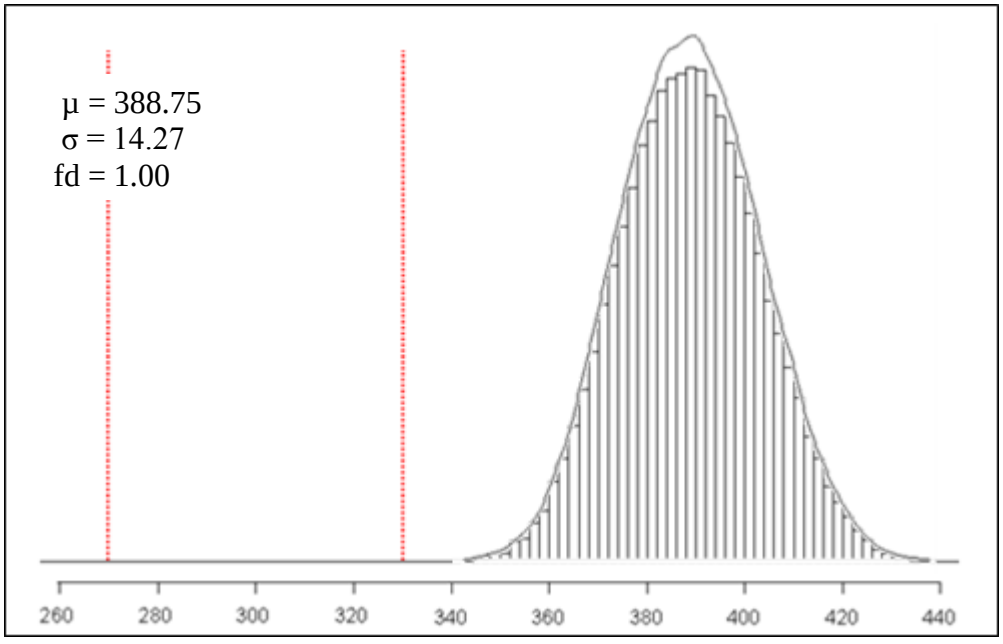


**Figura 6.** Histograma y densidad de la simulación basada en el diseño optimizado

Los resultados que se obtuvieron con la optimización Taguchi no son nada satisfactorios, si bien incremento la media del proceso respecto al diseño inicial esta se encuentra muy lejana respecto al valor objetivo y la varianza en el volumen inyectado no mostro cambios con respecto al primer diseño.

**b) Cálculo de los ICP y de los IC para el diseño optimizado con Taguchi**

En el Cuadro 15 se observa que el valor de cada uno de los ICP disminuyó considerablemente, en comparación con los valores que se observaron en el proceso inicial, de hecho se presentan valores negativos (Cuadro 3).

**Cuadro 15.** ICP y sus respectivos IC para el diseño optimizado

| | **ICP** | **LI** | **LS** |
|---|---|---|---|
| $C_p$ | 0.7024846 | 0.6456936 | 0.7649418 |
| $C_{p\_iso}$ | 0.7430041 | 0.7172276 | 0.7688092 |
| $C_{pk}$ | -1.3761135 | -1.5041190 | -1.2600243 |
| $C_{pk\_iso}$ | -1.4484637 | -1.5002924 | -1.3972662 |
| $C_{pm}$ | 0.1112333 | 0.1090414 | 0.1135014 |
| $C_{pm\_iso}$ | 0.1117289 | 0.1112745 | 0.1121749 |
| $C_{pmk}$ | -0.2178716 | -0.2202080 | -0.2154586 |
| $C_{pmk\_iso}$ | -0.2178105 | -0.2183163 | -0.2173034 |
| $C_s$ | -0.2172174 | -0.2195478 | -0.2140506 |
| $C_{s\_iso}$ | -0.2174971 | -0.2181334 | -0.2167351 |

**c) Análisis del modelo optimizado con Taguchi**

**Cuadro 16.** Nivel de confianza observado para cada uno de los ICP

| | ICP | ICP_iso |
|---|---|---|
| $C_p$ | 0.9503 | 0.9519 |
| $C_{pk}$ | 0.9508 | 0.9520 |
| $C_{pm}$ | 0.9516 | 0.9468 |
| $C_{pmk}$ | 0.9501 | 0.9493 |
| $C_s$ | 0.9455 | 0.9541 |

En el Cuadro 16 los niveles de confianza observados están alrededor del 95%, lo cual indica que los IC trabajan correctamente.

## 10. ANÁLISIS DE RESULTADOS

Comparando los tres métodos de optimización: Optimización aleatoria, Optimización NLM (Non Linear Minimization) y Optimización Taguchi, se encontró que:

La optimización NLM solo se desarrolla alrededor del primer máximo (o mínimo) local que encuentra. Esta estrategia de optimización solo se puede aprovechar cuando se conoce una región cercana al máximo (o mínimo) deseado. Con la optimización NLM se obtienen resultados semejantes a la optimización aleatoria, cuando se le proporciona la combinación de factores encontrada con la optimización aleatoria, como valores semilla. La variación del proceso disminuyó respecto al modelo inicial.

La metodología Taguchi que consiste en seleccionar el máximo señales a ruido (SN) si bien traslado a la media del proceso, no logró disminuir la fracción defectuosa. Esto se pudo haber logrado si en un principio se hubiese trabajado con más de dos niveles para cada factor, con objeto de identificar una curvatura que permitiera encontrar el máximo de SN. Por otro lado, trabajar con el máximo SN provocó que el proceso simulado se trasladara más allá de los límites de especificación, provocando valores negativos en los índices de capacidad (Cuadro 19). La variación del proceso no cambio respecto al modelo inicial. Esto implica que la metodología de Taguchi es limitada.

En contraste, en la optimización aleatoria se puede aplicar desde un mínimo de dos niveles por factor y además, como esta optimización recorre todo el espacio definido por los niveles de los factores, buscando combinaciones que se acerquen más al valor meta no se circunscribe a máximos locales. La variación del proceso disminuyó respecto al modelo inicial. Por lo anterior y tomando en consideración las medidas de eficiencia como la función de pérdida de Taguchi, los ICP y la desviación estándar del modelo se concluye que el mejor método es el de optimización aleatoria (Cuadro 17).

**Cuadro 17.** Diseño optimo

| Factor | | Diseño optimo | |
|---|---|---|---|
| | | Valor | Tolerancia |
| **A** | **Fuente de carga** | 749.31 | ± 50 |
| **B** | **Boquilla de flujo** | 9.02 | ± 0.15 |
| **C** | **Lanzadera de ascenso** | 0.28 | ± 0.03 |

En la Figura 7 se observa como la optimización aleatoria y la optimización NLM mejoraron la función de pérdida de Taguchi con respecto al diseño inicial. La función de perdida no se gráfico para la optimización aleatoria ya que algunos de los valores superaban los 10 millones.

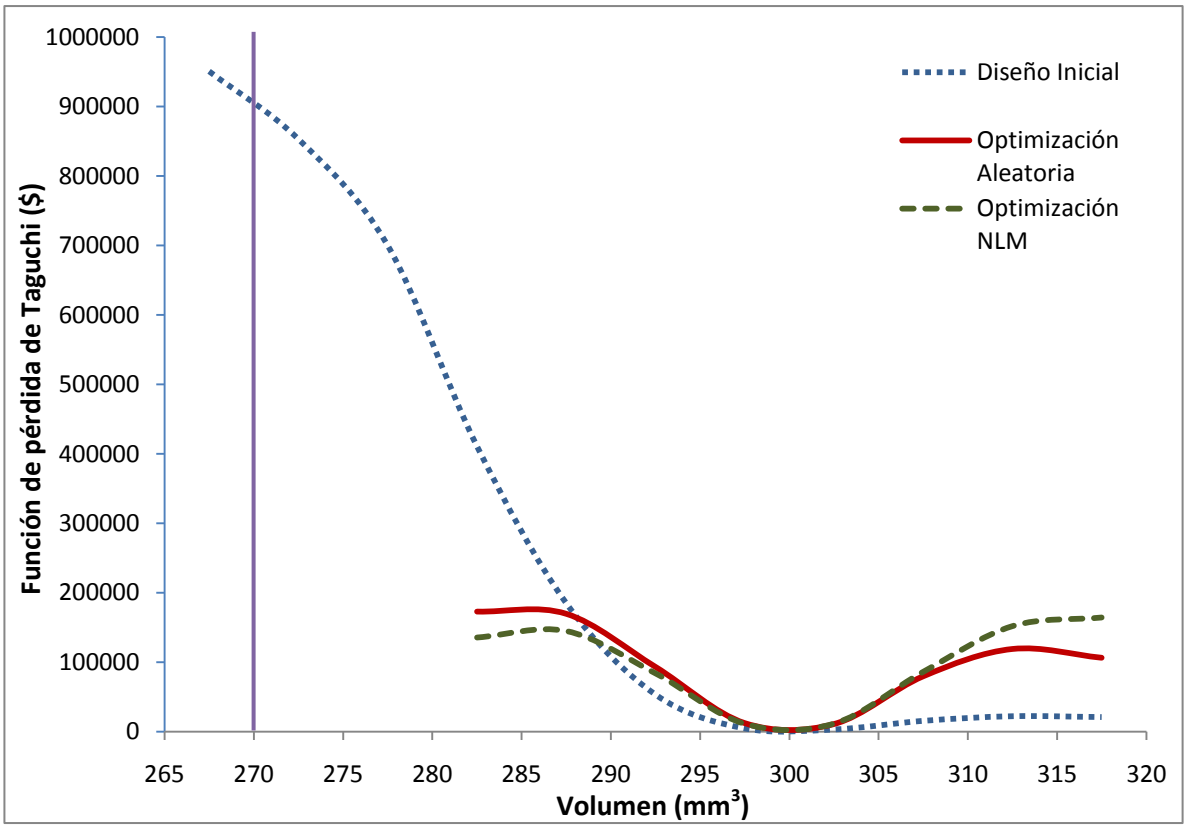


**Figura 7.** Grafica de la función de pérdida de Taguchi y el volumen inyectado

En el Cuadro 18 se observa que los diseños obtenidos con las estrategias de optimización aleatoria y NLM presentan valores más pequeños en el máximo de la función de pérdida de Taguchi respecto al modelo inicial.

Sin embargo, la optimización Taguchi deja mucho que desear, puesto que tiene el valor más grande del máximo de la función de pérdida de Taguchi comparada con las otras estrategias.

**Cuadro 18.** Media, desviación estándar del proceso y máximo de la función de pérdida de Taguchi del diseño inicial y los diferentes métodos de optimización.

| Diseño | Media | Desviación Estándar | L Taguchi (Máx.) |
|---|---|---|---|
| Optimización Taguchi | 388.75 | 14.27 | 11773400 |
| Inicial | 281.48 | 14.11 | 950625 |
| Optimización Aleatoria | 299.06 | 10.04 | 172725 |
| NLM | 300.05 | 10.27 | 164150 |

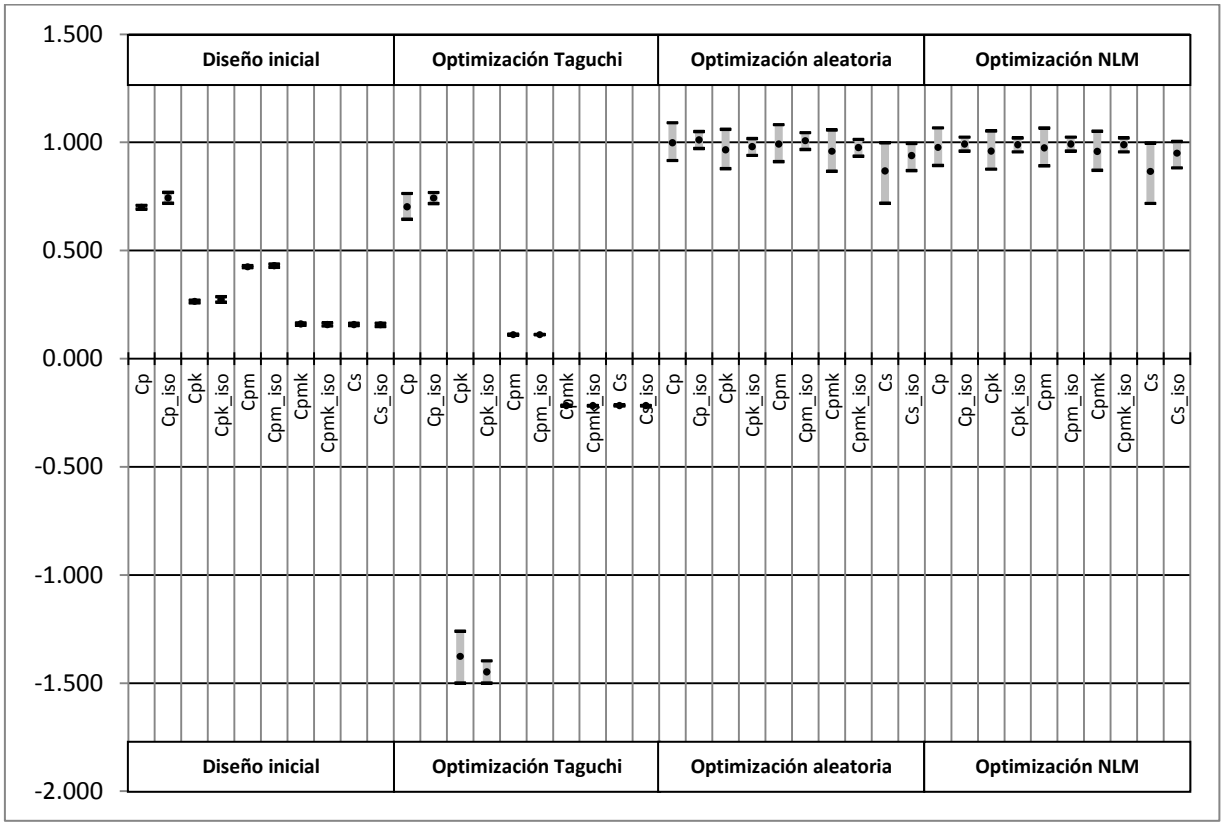


**Figura 8:** Gráfica de los ICP e IC de los métodos de optimización

Por medio de la metodología propuesta se logró identificar un diseño que mejoró el proceso (Cuadro 19, Figura 8) es decir, centro la media del proceso en el valor meta, redujo la

varianza del proceso, disminuyó la fracción defectuosa, se obtuvieron IC más angostos y el máximo de la función de perdida se redujo en comparación con el diseño inicial. Todo esto sin gastar en la producción de más prototipos.

**Cuadro 19.** IC de los diseños obtenidos con las rutinas de optimización y del diseño inicial.

| | **DISEÑO** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Inicial** | | | **O. Aleatoria** | | | **NLM** | | | **O. Taguchi** | | |
| | **ICP** | **LI** | **LS** | **ICP** | **LI** | **LS** | **ICP** | **LI** | **LS** | **ICP** | **LI** | **LS** |
| $C_p$ | 0.7007 | 0.6920 | 0.7095 | 0.9988 | 0.9168 | 1.0911 | 0.9770 | 0.8937 | 1.0676 | 0.7024 | 0.6456 | 0.7649 |
| $C_{p\_iso}$ | 0.7445 | 0.7197 | 0.7692 | 1.0128 | 0.9722 | 1.0500 | 0.9926 | 0.9606 | 1.0246 | 0.7430 | 0.7172 | 0.7688 |
| $C_{pk}$ | 0.2650 | 0.2582 | 0.2719 | 0.9660 | 0.8786 | 1.0609 | 0.9601 | 0.8763 | 1.0537 | -1.3761 | -1.5041 | -1.2600 |
| $C_{pk\_iso}$ | 0.2739 | 0.2614 | 0.2866 | 0.9812 | 0.9402 | 1.0182 | 0.9892 | 0.9566 | 1.0218 | -1.4484 | -1.5002 | -1.3972 |
| $C_{pm}$ | 0.4257 | 0.4216 | 0.4300 | 0.9923 | 0.9111 | 1.0826 | 0.9750 | 0.8928 | 1.0660 | 0.1112 | 0.1090 | 0.1135 |
| $C_{pm\_iso}$ | 0.4303 | 0.4231 | 0.4374 | 1.0082 | 0.9677 | 1.0450 | 0.9925 | 0.9605 | 1.0246 | 0.1117 | 0.1112 | 0.1121 |
| $C_{pmk}$ | 0.1610 | 0.1558 | 0.1663 | 0.9599 | 0.8673 | 1.0581 | 0.9582 | 0.8717 | 1.0527 | -0.2178 | -0.2202 | -0.2154 |
| $C_{pmk\_iso}$ | 0.1583 | 0.1512 | 0.1656 | 0.9767 | 0.9360 | 1.0141 | 0.9891 | 0.9565 | 1.0217 | -0.2178 | -0.2183 | -0.2173 |
| $C_s$ | 0.1587 | 0.1534 | 0.1641 | 0.8680 | 0.7192 | 0.9983 | 0.8661 | 0.7184 | 0.9978 | -0.2172 | -0.2195 | -0.2140 |
| $C_{s\_iso}$ | 0.1572 | 0.1499 | 0.1646 | 0.9398 | 0.8701 | 0.9956 | 0.9506 | 0.8820 | 1.0042 | -0.2174 | -0.2181 | -0.2167 |

## 11. CONCLUSIONES

Las rutinas de optimización aleatoria y NLM mejoran el proceso inicial, con la diferencia de que la optimización aleatoria no necesita de un valor inicial cercano al valor óptimo para no estancarse en un mínimo local.

La rutina NLM tiene la ventaja de ser más rápida computacionalmente en comparación con la optimización aleatoria, por lo que si se tiene experiencia con la función a optimizar y se sabe que no cuenta con óptimos locales puede emplearse para la optimización del proceso. De otro modo, si bien la optimización aleatoria necesita de un mayor trabajo computacional esta busca en toda la región a optimizar y no se detiene en óptimos locales lo cual da una mayor garantía de encontrar un óptimo global.

La optimización Taguchi necesita de al menos 3 niveles en cada factor para tener una idea más precisa del comportamiento de la curva SN y así maximizarla. Ya que al trabajar con 2 niveles para cada factor, se encontró que maximizando las SN provoca que el proceso simulado se trasladé más allá de los límites de especificación, provocando valores negativos en los índices de capacidad (Cuadro 19). Por otro lado, con la optimización Taguchi la variación del proceso no cambio respecto al modelo inicial.

En contraste, en la optimización aleatoria se puede aplicar desde un mínimo de dos niveles por factor y además, al considerar algunas medidas de eficiencia como la función de pérdida de Taguchi, los ICP y la desviación estándar del modelo se concluye que el mejor método es la optimización aleatoria (Cuadro 17).

Por medio de la metodología propuesta se logró identificar un diseño que mejoró el proceso (Cuadro 19, Figura 8) es decir, centro la media del proceso en el valor meta, redujo la varianza del proceso, disminuyó la fracción defectuosa, se obtuvieron IC más angostos en comparación con el diseño inicial y como se observó en la Figura 8 la función de pérdida de Taguchi se mejoro notablemente en comparación con el diseño inicial. Todo esto sin

gastar en la producción de más prototipos. Por lo que la metodología propuesta es una opción efectiva, flexible y robusta al optimizar procesos.

# 12. ANEXO

## Rutina 12.1 Ajuste del modelo con los datos obtenidos del diseño inicial

```
##########DISEÑO INICIAL##########

###Entrada de datos###

factores<-3
niveles<-2
replicas<-3

###Valor de cada uno de los factores###

Factor1<-c(500,900)
Factor2<-c(6,9)
Factor3<-c(0.3,0.6)

renglones<-replicas*(niveles^factores)
columnas<-(factores+1)
numceldas<-renglones*columnas

datos<-rep(0,numceldas)
Datos<-matrix(datos,renglones,columnas)

###Codificando los factores###

media1<-mean(Factor1)
sd1<-(Factor1[2]-Factor1[1])/2
media2<-mean(Factor2)
sd2<-(Factor2[2]-Factor2[1])/2
media3<-mean(Factor3)
sd3<-(Factor3[2]-Factor3[1])/2
cod1<-c((Factor1[1]-media1)/sd1,(Factor1[2]-media1)/sd1)
cod2<-c((Factor2[1]-media2)/sd2,(Factor2[2]-media2)/sd2)
cod3<-c((Factor3[1]-media3)/sd3,(Factor3[2]-media3)/sd3)

Datos[,1]<-
c(126,141,122,183,168,164,284,283,275,300,318,310,249,242,242,125,128,140,387,392,391,284,26
9,255)
Datos[,2]<-
c(cod1[1],cod1[1],cod1[1],cod1[2],cod1[2],cod1[2],cod1[1],cod1[1],cod1[1],cod1[2],cod1[2],co
d1[2],cod1[1],cod1[1],cod1[1],cod1[2],cod1[2],cod1[2],cod1[1],cod1[1],cod1[1],cod1[2],cod1[2
],cod1[2])
Datos[,3]<-
c(cod2[1],cod2[1],cod2[1],cod2[1],cod2[1],cod2[1],cod2[2],cod2[2],cod2[2],cod2[2],cod2[2],co
d2[2],cod2[1],cod2[1],cod2[1],cod2[1],cod2[1],cod2[1],cod2[2],cod2[2],cod2[2],cod2[2],cod2[2
],cod2[2])
Datos[,4]<-
c(cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],cod3[1],co
d3[1],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2],cod3[2
],cod3[2])

Datosfactorial<-as.data.frame(Datos)

LinearModel.1 <- lm(V1  ~ V2  + V3  + V4  + V2 *V3 + V2*V4 + V3*V4 + V2*V3*V4 ,
data=Datosfactorial)

###En el modelo anterior observamos que solo son significativos V2 V3 V4 y V2*V4,###
###así el modelo a trabajar es el siguiente:###

LinearModel.2 <- lm(V1  ~ V2  + V3  + V4  + V2 *V4, data=Datosfactorial)
summary(LinearModel.1)
summary(LinearModel.2)
```

## Rutina 12.1.1 Simulación Monte Carlo del diseño inicial

```
##########ANALISIS USANSO SIMULACION MONTE CARLO##########

###Simulación Monte Carlo (Modelo Inicial)###

###Entrada de datos (número de replicas y límites de especificación)###

r<-100000
usl<-330
lsl<-270

###Niveles de los Factores###

Factor1<-c(500,900)
Factor2<-c(6,9)
Factor3<-c(0.3,0.6)

###Valores del diseño inicial###

Fac_A<-c(450,550)
Fac_B<-c(6.6,6.9)
Fac_C<-c(0.57,0.63)

###Codificación de los valores del diseño inicial###

F_A<-c((Fac_A[1]-mean(Factor1))/(mean(Factor1)-Factor1[1]),(Fac_A[2]-
mean(Factor1))/(mean(Factor1)-Factor1[1]))
F_B<-c((Fac_B[1]-mean(Factor2))/(mean(Factor2)-Factor2[1]),(Fac_B[2]-
mean(Factor2))/(mean(Factor2)-Factor2[1]))
F_C<-c((Fac_C[1]-mean(Factor3))/(mean(Factor3)-Factor3[1]),(Fac_C[2]-
mean(Factor3))/(mean(Factor3)-Factor3[1]))

###Amplitudes codificadas de cada uno de los factores del diseño inicial###

AFAC<-F_A[2]-F_A[1]
AFBC<-F_B[2]-F_B[1]
AFCC<-F_C[2]-F_C[1]

###Simulación de un inyector###

Voli<-function()
               { repeat
                      {###Entrada de datos del modelo inicial###

                             Intercepto<-240.750
                             Coef_A<--20.417
                             Coef_B<-71.583
                             Coef_C<-17.917
                             Coef_AC<--38.083

                       ###Simulacion de valores para A B C AC respectivamente###

                             A<-runif(1,F_A[1],F_A[2])
                             B<-runif(1,F_B[1],F_B[2])
                             C<-runif(1,F_C[1],F_C[2])
                             AC<-A*C

                             Variacion<-rnorm(1,0,8.547)

                       ###Simulacion de un valor de Y (Volumen)###

              Y<-Intercepto + Coef_A*A + Coef_B*B + Coef_C*C + Coef_AC*AC + Variacion

                       ###Valores que guarda el programa###

                             vol<-Y
                             return(vol)

                      }
               }
```

```
Vol<-Voli()

###La siguiente rutina implementa al programa para que proporcione r replicas simuladas de Y
(Volumen)###

vol<-rep(0,r)

for (i in 1:r)
        {
        vol[i]<-Voli()
        }

###Calculo de la Fracción Defectuosa (fd)###

fc_<-which(vol>=lsl & vol<=usl)
fc<-length(fc_)/r
fd<-1-fc
fd

###Media y desviación estándar del volumen inyectado por las unidades simuladas (mvol y
sdvol)###

mvol<-mean(vol)
sdvol<-sd(vol)
mvol
sdvol

###Histograma y densidad empírica del volumen inyectado por las unidades simuladas###

hist(vol, breaks=40)
plot(density(vol))
```

## Rutina 12.1.2 Intervalos de confianza para los ICP (Diseño inicial)

```
##########Intervalos de Confianza para los ICP (Datos del diseño inicial)##########

###Datos del programa de simulación del volumen (Diseño Inicial)###

datosvol<-vol
tolrange<-usl-lsl
r<-10000
n<-250

###Rutina para construir los ICP###

ICPi=function()
               {repeat
                       {###Remuestreo de los datos###

                        x<-sample(datosvol,r,TRUE)
                       x2<-sample(x,n,TRUE)

                       ###Calculo de los cuantiles###

                       LI<-quantile(x,0.00135)
                       MA<-quantile(x,0.5)
                       LS<-quantile(x,0.99865)

                       ###Media y desviación estándar de los datos###

                            mx<-mean(x2)
                           sdx<-sd(x2)

                       ###Construcción del Índice Cp (normal, no-normal)###

                          I_Cp<-(tolrange/(6*(sdx)))
                         I_Cp2<-(tolrange/((LS-LI)))

                       ###Construcción del Índice Cpk (normal, no-normal)###
```

```
                              d<-tolrange/2
                              M<-(usl+lsl)/2
                            I_Cpk<-(d-abs(mx-M))/(3*sdx)
                           I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                        ###Construcción del Índice Cpm (normal, no-normal)###

                              T<-M
                            I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                           I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

      ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                           I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                          I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                        ###Construcción del índice Cs (normal, no-normal)###

                            ite<-n
                    estadistico<-rep(0,ite)
                                       for(i in 1:ite)
                                       {
                                       estadistico[i]<-(x[i]-mx)^3
                                       }
                             mu_3<-mean(estadistico)

                             I_Cs<-(d-abs(mx-M))/(3*sqrt(sdx^2+(mx-T)^2+abs(mu_3/sdx)))
                            I_Cs2<-(d-abs(MA-M))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+abs(mu_3/d)))

                        ###Vector que contiene los valores de los ICP###

                     I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                        ###Este procedimiento ahora nos proporciona los valores de los ICP###

                        return(I)

                               }
                        }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###
ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
       {
       indices[i,]<-ICPi()
       }

###Construcción de los Intervalos de Confianza###

###Estimador bootstrap del Índice Cp (normal, no-normal)###

mediaI_Cp<-mean(indices[,1])
mediaI_Cp2<-mean(indices[,2])

###IC para los índices Cp (normal, no-normal)###

indicesI_Cp<-indices[,1]
IC_I_Cp<-c(quantile(indicesI_Cp,0.025),quantile(indicesI_Cp,0.975))

indicesI_Cp2<-indices[,2]
IC_I_Cp2<-c(quantile(indicesI_Cp2,0.025),quantile(indicesI_Cp2,0.975))

###Estimador bootstrap del Índice Cpk (normal, no-normal)###
```

```
mediaI_Cpk<-mean(indices[,3])
mediaI_Cpk2<-mean(indices[,4])

###IC para los índices Cpk (normal, no-normal)###

indicesI_Cpk<-indices[,3]
IC_I_Cpk<-c(quantile(indicesI_Cpk,0.025),quantile(indicesI_Cpk,0.975))

indicesI_Cpk2<-indices[,4]
IC_I_Cpk2<-c(quantile(indicesI_Cpk2,0.025),quantile(indicesI_Cpk2,0.975))

###Estimador bootstrap del Índice Cpm (normal, no-normal)###

mediaI_Cpm<-mean(indices[,5])
mediaI_Cpm2<-mean(indices[,6])

###IC para los índices Cpm (normal, no-normal)###

indicesI_Cpm<-indices[,5]
IC_I_Cpm<-c(quantile(indicesI_Cpm,0.025),quantile(indicesI_Cpm,0.975))

indicesI_Cpm2<-indices[,6]
IC_I_Cpm2<-c(quantile(indicesI_Cpm2,0.025),quantile(indicesI_Cpm2,0.975))

####Estimador bootstrap del Índice Cpmk (normal, no-normal)###

mediaI_Cpmk<-mean(indices[,7])
mediaI_Cpmk2<-mean(indices[,8])

###IC para los índices Cpmk (normal, no-normal)###

indicesI_Cpmk<-indices[,7]
IC_I_Cpmk<-c(quantile(indicesI_Cpmk,0.025),quantile(indicesI_Cpmk,0.975))

indicesI_Cpmk2<-indices[,8]
IC_I_Cpmk2<-c(quantile(indicesI_Cpmk2,0.025),quantile(indicesI_Cpmk2,0.975))

####Estimador bootstrap del Índice Cs (normal, no-normal)###

mediaI_Cs<-mean(indices[,9])
mediaI_Cs2<-mean(indices[,10])

###IC para los índices Cs (normal, no-normal)###

indicesI_Cs<-indices[,9]
IC_I_Cs<-c(quantile(indicesI_Cs,0.025),quantile(indicesI_Cs,0.975))

indicesI_Cs2<-indices[,10]
IC_I_Cs2<-c(quantile(indicesI_Cs2,0.025),quantile(indicesI_Cs2,0.975))


###SALIDA (Indice de Capacidad, Intervalo de Confianza al 95% (LI, LS))###

valor<-rep(0,ni*3)
valores<-matrix(valor,ni,3)

valores[1,1]<-mediaI_Cp
valores[1,2]<-IC_I_Cp[1]
valores[1,3]<-IC_I_Cp[2]

valores[2,1]<-mediaI_Cp2
valores[2,2]<-IC_I_Cp2[1]
valores[2,3]<-IC_I_Cp2[2]

valores[3,1]<-mediaI_Cpk
valores[3,2]<-IC_I_Cpk[1]
valores[3,3]<-IC_I_Cpk[2]

valores[4,1]<-mediaI_Cpk2
valores[4,2]<-IC_I_Cpk2[1]
valores[4,3]<-IC_I_Cpk2[2]
```

```
valores[5,1]<-mediaI_Cpm
valores[5,2]<-IC_I_Cpm[1]
valores[5,3]<-IC_I_Cpm[2]

valores[6,1]<-mediaI_Cpm2
valores[6,2]<-IC_I_Cpm2[1]
valores[6,3]<-IC_I_Cpm2[2]

valores[7,1]<-mediaI_Cpmk
valores[7,2]<-IC_I_Cpmk[1]
valores[7,3]<-IC_I_Cpmk[2]

valores[8,1]<-mediaI_Cpmk2
valores[8,2]<-IC_I_Cpmk2[1]
valores[8,3]<-IC_I_Cpmk2[2]

valores[9,1]<-mediaI_Cs
valores[9,2]<-IC_I_Cs[1]
valores[9,3]<-IC_I_Cs[2]

valores[10,1]<-mediaI_Cs2
valores[10,2]<-IC_I_Cs2[1]
valores[10,3]<-IC_I_Cs2[2]

valores
```

## Rutina 12.1.3 Porcentajes de cobertura de los IC (Diseño inicial)

```
##########ANALISIS DE LOS PORCENTAJES DE COBERTURA##########

###Valores de los Intervalos de Confianza (Diseño Inicial)###

IC_I_Cp
IC_I_Cp2
IC_I_Cpk
IC_I_Cpk2
IC_I_Cpm
IC_I_Cpm2
IC_I_Cpmk
IC_I_Cpmk2
IC_I_Cs
IC_I_Cs2

###DATOS del programa de simulación del volumen (Diseño Inicial)###

datosvol<-vol
r<-10000
n<-250

ICPi=function()
               {repeat
                      {###Remuestreo de los datos###

                       x<-sample(datosvol,r,TRUE)
                      x2<-sample(x,n,TRUE)

                       ###Calculo de los cuantiles###

                      LI<-quantile(x,0.00135)
                      MA<-quantile(x,0.5)
                      LS<-quantile(x,0.99865)

                      ###Media y desviación estándar de los datos###

                      mx<-mean(x2)
                     sdx<-sd(x2)

                      ###Construcción del Índice Cp (normal, no-normal)###
```

```
                        I_Cp<-(tolrange/(6*(sdx)))
                       I_Cp2<-(tolrange/(LS-LI))

                     ###Construcción del Índice Cpk (normal, no-normal)###

                            d<-tolrange/2
                            M<-(usl+lsl)/2
                       I_Cpk<-(d-abs(mx-M))/(3*sdx)
                      I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                     ###Construcción del Índice Cpm (normal, no-normal)###

                           T<-M
                       I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                      I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

      ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                      I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                     I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                     ###Construcción del índice Cs (normal, no-normal)###

                     ###Calculo del tercer momento de la variable vol###

                        mux3<-mean(x2)
                         ite<-n
               estadistico<-rep(0,ite)
                                  for(i in 1:ite)
                                  {
                                  estadistico[i]<-(x2[i]-mux3)^3
                                  }
                        mu_3<-mean(estadistico)

             I_Cs<-(d-(abs(mx-M)))/(3*sqrt(sdx^2+(mx-T)^2+(abs(mu_3/sdx))))
            I_Cs2<-(d-(abs(MA-M)))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+(abs(mu_3/d))))

                     ###Vector que contiene los valores de los ICP###

                I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                     ###Este procedimiento ahora nos proporciona los valores de los ICP###

                     return(I)

                            }
                     }

ICP<-ICPi()

###El siguiente código implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###
ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
       {
       indices[i,]<-ICPi()
       }

###Calculo del porcentaje de cobertura de los ICP###

pCp<-which(indices[,1]>=IC_I_Cp[1] & indices[,1]<=IC_I_Cp[2])
p_Cp<-length(pCp)/r

pCp2<-which(indices[,2]>=IC_I_Cp2[1] & indices[,2]<=IC_I_Cp2[2])
p_Cp2<-length(pCp2)/r

pCpk<-which(indices[,3]>=IC_I_Cpk[1] & indices[,3]<=IC_I_Cpk[2])
p_Cpk<-length(pCpk)/r
```

```
pCpk2<-which(indices[,4]>=IC_I_Cpk2[1] & indices[,4]<=IC_I_Cpk2[2])
p_Cpk2<-length(pCpk2)/r

pCpm<-which(indices[,5]>=IC_I_Cpm[1] & indices[,5]<=IC_I_Cpm[2])
p_Cpm<-length(pCpm)/r

pCpm2<-which(indices[,6]>=IC_I_Cpm2[1] & indices[,6]<=IC_I_Cpm2[2])
p_Cpm2<-length(pCpm2)/r

pCpmk<-which(indices[,7]>=IC_I_Cpmk[1] & indices[,7]<=IC_I_Cpmk[2])
p_Cpmk<-length(pCpmk)/r

pCpmk2<-which(indices[,8]>=IC_I_Cpmk2[1] & indices[,8]<=IC_I_Cpmk2[2])
p_Cpmk2<-length(pCpmk2)/r

pCs<-which(indices[,9]>=IC_I_Cs[1] & indices[,9]<=IC_I_Cs[2])
p_Cs<-length(pCs)/r

pCs2<-which(indices[,10]>=IC_I_Cs2[1] & indices[,10]<=IC_I_Cs2[2])
p_Cs2<-length(pCs2)/r

###Porcentajes de cobertura###

pc<-c(p_Cp,p_Cpk,p_Cpm,p_Cpmk,p_Cs,p_Cp2,p_Cpk2,p_Cpm2,p_Cpmk2,p_Cs2)

PC<-matrix(pc,ni/2,2)

###Salida del Programa, en la 1er columna se observa el valor del ICP, en la 2da y 3ra se
observan el límite inferior y superior del ICP.###

PC
```

## Rutina 12.2 Optimización aleatoria

```
##########OPTIMIZACION ALEATORIA##########

###Valores iniciales###

dif<-0
aaa<-c(-1,1)
bbb<-c(-1,1)
ccc<-c(-1,1)
usl<-330
lsl<-270

###Niveles de los Factores###

Factor1<-c(500,900)
Factor2<-c(6,9)
Factor3<-c(0.3,0.6)

###Valores del diseño inicial###

Fac_A<-c(450,550)
Fac_B<-c(6.6,6.9)
Fac_C<-c(0.57,0.63)

###Codificación de los valores del diseño inicial###

F_A<-c((Fac_A[1]-mean(Factor1))/(mean(Factor1)-Factor1[1]),(Fac_A[2]-
mean(Factor1))/(mean(Factor1)-Factor1[1]))
F_B<-c((Fac_B[1]-mean(Factor2))/(mean(Factor2)-Factor2[1]),(Fac_B[2]-
mean(Factor2))/(mean(Factor2)-Factor2[1]))
F_C<-c((Fac_C[1]-mean(Factor3))/(mean(Factor3)-Factor3[1]),(Fac_C[2]-
mean(Factor3))/(mean(Factor3)-Factor3[1]))

###Amplitudes codificadas de cada uno de los factores del diseño inicial###

AFAC<-F_A[2]-F_A[1]
```

```
AFBC<-F_B[2]-F_B[1]
AFCC<-F_C[2]-F_C[1]

###Inicio de la rutina###

                        dif<-dif
                        aaa<-aaa
                        bbb<-bbb
                        ccc<-ccc

Voli<-function()
              { repeat
                     {###Entrada de datos###

                            Intercepto<-240.750
                            Coef_A<--20.417
                            Coef_B<-71.583
                            Coef_C<-17.917
                            Coef_AC<--38.083

                     ###Simulación de valores para A B C AC respectivamente###

                            A<-runif(1,aaa[1],aaa[2])
                            B<-runif(1,bbb[1],bbb[2])
                            C<-runif(1,ccc[1],ccc[2])
                            AC<-A*C

                            Variacion<-rnorm(1,0,8.547)

                     ###Simulación de un valor de Y (Volumen)###

              Y<-Intercepto + Coef_A*A + Coef_B*B + Coef_C*C + Coef_AC*AC + Variacion

                     ###Optimización aleatoria###

                            loss<-(Y-300)^2

                     ###Valores que guarda el programa###

                            ABCYloss<-c(A,B,C,Y,loss)
                            return(ABCYloss)

                     }
              }

Vol<-Voli()

###El siguiente código implementa al programa para que proporcione rop replicas de Y
(Volumen)###

vs<-5
rop<-10000
salida_<-rep(0,vs*rop)
salida<-matrix(salida_,rop,vs)

for (i in 1:rop)
       {
       salida[i,]<-Voli()
       }

###Optimización###

mloss<-min(salida[,vs])
opt<-which(salida[,vs]==mloss)
optim<-salida[c(opt),]

###Valores óptimos codificados###

aaa<-c(optim[1]-AFAC/2,optim[1]+AFAC/2)
bbb<-c(optim[2]-AFBC/2,optim[2]+AFBC/2)
ccc<-c(optim[3]-AFCC/2,optim[3]+AFCC/2)
```

```
###SIMULACION MONTECARLO DEL DISEÑO OPTIMO (Optimización Aleatoria)###

Volui<-function()
               { repeat
                      {###Entrada de datos###

                             Intercepto<-240.750
                             Coef_A<--20.417
                             Coef_B<-71.583
                             Coef_C<-17.917
                             Coef_AC<--38.083

                      ###Simulacion de valores para A B C AC respectivamente###

                             A<-runif(1,aaa[1],aaa[2])
                             B<-runif(1,bbb[1],bbb[2])
                             C<-runif(1,ccc[1],ccc[2])
                             AC<-A*C

                             Variacion<-rnorm(1,0,8.547)

                      ###Simulacion de un valor de Y (Volumen)###

              Y<-Intercepto + Coef_A*A + Coef_B*B + Coef_C*C + Coef_AC*AC + Variacion

                      ###Valores que guarda el programa###

                             Yoptim<-Y
                             return(Yoptim)

                      }
               }

Volu<-Volui()

###La siguiente rutina implementa al programa para que nos proporcione r replicas de Y
(Volumen)###

r<-10000

volu<-rep(0,r)

for (i in 1:r)
       {
       volu[i]<-Volui()
       }

###Calculo de la media y desviación estándar del proceso###

mvol<-mean(volu)
desviacion<-sd(volu)

###Calculo de la probabilidad de cumplir con las especificaciones###

p_<-which(volu>=lsl & volu<=usl)
p<-length(p_)/r

###Calculo de la fracción disconforme (fd)###

fd<-1-p

###Valores optimos sin codificar###

Ao<-(mean(aaa)*(Factor1[2]-mean(Factor1)))+mean(Factor1)
Bo<-(mean(bbb)*(Factor2[2]-mean(Factor2)))+mean(Factor2)
Co<-(mean(ccc)*(Factor3[2]-mean(Factor3)))+mean(Factor3)

valores_A_B_C_Optimos<-c(Ao,Bo,Co)
valores_A_B_C_Optimos
```

```
###Media del proceso###

mvol

###Desviación estándar del proceso###

desviacion

###Fracción disconforme###

fd

###Fracción no disconforme (Probabilidad de cumplir con las especificaciones)###

p

###Histograma y densidad empírica del volumen inyectado por las unidades simuladas###

hist(volu, breaks=40)
plot(density(volu))
```

## Rutina 12.2.1 Intervalos de confianza para los ICP (Optimización aleatoria)

```
###DATOS del programa de la simulación Monte Carlo del Diseño Optimo###

datosvol<-volu
tolrange<-usl-lsl

###Datos para el remuestreo bootstrap###

r<-10000
n<-250

ICPi=function()
               {repeat
                       {###Remuestreo de los datos###

                        x<-sample(datosvol,r,TRUE)
                       x2<-sample(x,n,TRUE)

                        ###Calculo de los cuantiles###

                       LI<-quantile(x,0.00135)
                       MA<-quantile(x,0.5)
                       LS<-quantile(x,0.99865)

                       ###Media y desviación estándar de los datos###

                            mx<-mean(x2)
                           sdx<-sd(x2)

                       ###Construcción del Índice Cp (normal, no-normal)###

                          I_Cp<-(tolrange/(6*(sdx)))
                         I_Cp2<-(tolrange/((LS-LI)))

                       ###Construcción del Índice Cpk (normal, no-normal)###

                               d<-tolrange/2
                               M<-(usl+lsl)/2
                          I_Cpk<-(d-abs(mx-M))/(3*sdx)
                         I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                       ###Construcción del Índice Cpm (normal, no-normal)###

                             T<-M
                         I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                        I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))
```

```
        ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                        I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                       I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                       ###Construcción del índice Cs (normal, no-normal)###

                           ite<-n
                    estadistico<-rep(0,ite)
                                       for(i in 1:ite)
                                       {
                                       estadistico[i]<-(x[i]-mx)^3
                                       }
                              mu_3<-mean(estadistico)

                       I_Cs<-(d-abs(mx-M))/(3*sqrt(sdx^2+(mx-T)^2+abs(mu_3/sdx)))
                       I_Cs2<-(d-abs(MA-M))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+abs(mu_3/d)))

                       ###Vector que contiene los valores de los ICP###

              I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                       return(I)

                              }
                       }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###
ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
       {
       indices[i,]<-ICPi()
       }

###Construcción de los Intervalos de Confianza###

###Estimador bootstrap del Índice Cp (normal, no-normal)###

mediaI_Cp<-mean(indices[,1])
mediaI_Cp2<-mean(indices[,2])

###IC para los índices Cp (normal, no-normal)###

indicesI_Cp<-indices[,1]
IC_I_Cp<-c(quantile(indicesI_Cp,0.025),quantile(indicesI_Cp,0.975))

indicesI_Cp2<-indices[,2]
IC_I_Cp2<-c(quantile(indicesI_Cp2,0.025),quantile(indicesI_Cp2,0.975))

###Estimador bootstrap del Índice Cpk (normal, no-normal)###

mediaI_Cpk<-mean(indices[,3])
mediaI_Cpk2<-mean(indices[,4])

###IC para los índices Cpk (normal, no-normal)###

indicesI_Cpk<-indices[,3]
IC_I_Cpk<-c(quantile(indicesI_Cpk,0.025),quantile(indicesI_Cpk,0.975))

indicesI_Cpk2<-indices[,4]
IC_I_Cpk2<-c(quantile(indicesI_Cpk2,0.025),quantile(indicesI_Cpk2,0.975))

###Estimador bootstrap del Índice Cpm (normal, no-normal)###

mediaI_Cpm<-mean(indices[,5])
```

```
mediaI_Cpm2<-mean(indices[,6])

###IC para los índices Cpm (normal, no-normal)###

indicesI_Cpm<-indices[,5]
IC_I_Cpm<-c(quantile(indicesI_Cpm,0.025),quantile(indicesI_Cpm,0.975))

indicesI_Cpm2<-indices[,6]
IC_I_Cpm2<-c(quantile(indicesI_Cpm2,0.025),quantile(indicesI_Cpm2,0.975))

###Estimador bootstrap del Índice Cpmk (normal, no-normal)###

mediaI_Cpmk<-mean(indices[,7])
mediaI_Cpmk2<-mean(indices[,8])

###IC para los índices Cpmk (normal, no-normal)###

indicesI_Cpmk<-indices[,7]
IC_I_Cpmk<-c(quantile(indicesI_Cpmk,0.025),quantile(indicesI_Cpmk,0.975))

indicesI_Cpmk2<-indices[,8]
IC_I_Cpmk2<-c(quantile(indicesI_Cpmk2,0.025),quantile(indicesI_Cpmk2,0.975))

###Estimador bootstrap del Índice Cs (normal, no-normal)###

mediaI_Cs<-mean(indices[,9])
mediaI_Cs2<-mean(indices[,10])

###IC para los indices Cs (normal, no-normal)###

indicesI_Cs<-indices[,9]
IC_I_Cs<-c(quantile(indicesI_Cs,0.025),quantile(indicesI_Cs,0.975))

indicesI_Cs2<-indices[,10]
IC_I_Cs2<-c(quantile(indicesI_Cs2,0.025),quantile(indicesI_Cs2,0.975))


###SALIDA: [Índice de Capacidad, Intervalo de Confianza al 95% (LI, LS)]###

valor<-rep(0,ni*3)
valores<-matrix(valor,ni,3)

valores[1,1]<-mediaI_Cp
valores[1,2]<-IC_I_Cp[1]
valores[1,3]<-IC_I_Cp[2]

valores[2,1]<-mediaI_Cp2
valores[2,2]<-IC_I_Cp2[1]
valores[2,3]<-IC_I_Cp2[2]

valores[3,1]<-mediaI_Cpk
valores[3,2]<-IC_I_Cpk[1]
valores[3,3]<-IC_I_Cpk[2]

valores[4,1]<-mediaI_Cpk2
valores[4,2]<-IC_I_Cpk2[1]
valores[4,3]<-IC_I_Cpk2[2]

valores[5,1]<-mediaI_Cpm
valores[5,2]<-IC_I_Cpm[1]
valores[5,3]<-IC_I_Cpm[2]

valores[6,1]<-mediaI_Cpm2
valores[6,2]<-IC_I_Cpm2[1]
valores[6,3]<-IC_I_Cpm2[2]

valores[7,1]<-mediaI_Cpmk
valores[7,2]<-IC_I_Cpmk[1]
valores[7,3]<-IC_I_Cpmk[2]

valores[8,1]<-mediaI_Cpmk2
```

```
valores[8,2]<-IC_I_Cpmk2[1]
valores[8,3]<-IC_I_Cpmk2[2]

valores[9,1]<-mediaI_Cs
valores[9,2]<-IC_I_Cs[1]
valores[9,3]<-IC_I_Cs[2]

valores[10,1]<-mediaI_Cs2
valores[10,2]<-IC_I_Cs2[1]
valores[10,3]<-IC_I_Cs2[2]

valores
```

## Rutina 12.2.2 Porcentajes de cobertura de los IC (Optimización aleatoria)

```
##########ANALISIS DE LOS PORCENTAJES DE COBERTURA (Optimización Aleatoria)##########

###Valores de los Intervalos de Confianza Obtenidos con Optimización Aleatoria###

IC_I_Cp
IC_I_Cp2
IC_I_Cpk
IC_I_Cpk2
IC_I_Cpm
IC_I_Cpm2
IC_I_Cpmk
IC_I_Cpmk2
IC_I_Cs
IC_I_Cs2

###DATOS del programa de simulación del volumen (Optimización Aleatoria)###

datosvol<-volu
r<-10000
n<-250

ICPi=function()
              {repeat
                     {###Remuestreo de los datos###

                       x<-sample(datosvol,r,TRUE)
                     x2<-sample(x,n,TRUE)

                      ###Calculo de los cuantiles###

                     LI<-quantile(x,0.00135)
                     MA<-quantile(x,0.5)
                     LS<-quantile(x,0.99865)

                     ###Media y desviación estándar de los datos###

                          mx<-mean(x2)
```

```
                    sdx<-sd(x2)

                ###Construcción del Índice Cp (normal, no-normal)###

                   I_Cp<-(tolrange/(6*(sdx)))
                  I_Cp2<-(tolrange/(LS-LI))

                ###Construcción del Índice Cpk (normal, no-normal)###

                        d<-tolrange/2
                        M<-(usl+lsl)/2
                  I_Cpk<-(d-abs(mx-M))/(3*sdx)
                 I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                ###Construcción del Índice Cpm (normal, no-normal)###

                      T<-M
                  I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                 I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                 I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                ###Construcción del índice Cs (normal, no-normal)###

               ###Calculo del tercer momento de la variable vol###

                   mux3<-mean(x2)
                    ite<-n
          estadistico<-rep(0,ite)
                              for(i in 1:ite)
                              {
                              estadistico[i]<-(x2[i]-mux3)^3
                              }
                   mu_3<-mean(estadistico)
        I_Cs<-(d-(abs(mx-M)))/(3*sqrt(sdx^2+(mx-T)^2+(abs(mu_3/sdx))))
        I_Cs2<-(d-(abs(MA-M)))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+(abs(mu_3/d))))

                ###Vector que contiene los valores de los ICP###

            I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                return(I)

                       }
                }
```

```
ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###

ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
        {
        indices[i,]<-ICPi()
        }

###Calculo del porcentaje de cobertura de los ICP###

pCp<-which(indices[,1]>=IC_I_Cp[1] & indices[,1]<=IC_I_Cp[2])
p_Cp<-length(pCp)/r

pCp2<-which(indices[,2]>=IC_I_Cp2[1] & indices[,2]<=IC_I_Cp2[2])
p_Cp2<-length(pCp2)/r

pCpk<-which(indices[,3]>=IC_I_Cpk[1] & indices[,3]<=IC_I_Cpk[2])
p_Cpk<-length(pCpk)/r

pCpk2<-which(indices[,4]>=IC_I_Cpk2[1] & indices[,4]<=IC_I_Cpk2[2])
p_Cpk2<-length(pCpk2)/r

pCpm<-which(indices[,5]>=IC_I_Cpm[1] & indices[,5]<=IC_I_Cpm[2])
p_Cpm<-length(pCpm)/r

pCpm2<-which(indices[,6]>=IC_I_Cpm2[1] & indices[,6]<=IC_I_Cpm2[2])
p_Cpm2<-length(pCpm2)/r

pCpmk<-which(indices[,7]>=IC_I_Cpmk[1] & indices[,7]<=IC_I_Cpmk[2])
p_Cpmk<-length(pCpmk)/r

pCpmk2<-which(indices[,8]>=IC_I_Cpmk2[1] & indices[,8]<=IC_I_Cpmk2[2])
p_Cpmk2<-length(pCpmk2)/r

pCs<-which(indices[,9]>=IC_I_Cs[1] & indices[,9]<=IC_I_Cs[2])
p_Cs<-length(pCs)/r

pCs2<-which(indices[,10]>=IC_I_Cs2[1] & indices[,10]<=IC_I_Cs2[2])
p_Cs2<-length(pCs2)/r

###Porcentajes de cobertura###
```

```
pc<-c(p_Cp,p_Cpk,p_Cpm,p_Cpmk,p_Cs,p_Cp2,p_Cpk2,p_Cpm2,p_Cpmk2,p_Cs2)

PC<-matrix(pc,ni/2,2)

PC
```

## Rutina 12.3 Optimización NLM

```
###Numero de replicas para la simulación Monte Carlo y limites de especificación###

r<-100000
lsl<-270
usl<-330

###Niveles de los Factores###

Factor1<-c(500,900)
Factor2<-c(6,9)
Factor3<-c(0.3,0.6)

###Optimización usando la función NLM de R###

fy<-function(y)
{
a<-y[1]
b<-y[2]
c<-y[3]

return((300-(240.75-20.42*a+71.58*b+17.92*c-38.08*a*c))^2)
}

###Valores semilla (Valores obtenidos en la optimización aleatoria)###
anlm<-aaa
bnlm<-bbb
cnlm<-ccc

nlm(fy,c(mean(anlm),mean(bnlm),mean(cnlm)))

###Valores capturados de la salida de la optimización NLM###

A_onlm<--0.4158812
B_onlm<-0.9328435
C_onlm<--0.4744313

###Calculo de los valores para realizar los valores de la simulación Monte Carlo###

afa<-aaa[2]-mean(anlm)
```

```
afb<-bbb[2]-mean(bnlm)
afc<-ccc[2]-mean(cnlm)

###Valores óptimos sin codificar###

A_onlm_sc<-(A_onlm*(Factor1[2]-mean(Factor1)))+mean(Factor1)
B_onlm_sc<-(B_onlm*(Factor2[2]-mean(Factor2)))+mean(Factor2)
C_onlm_sc<-(C_onlm*(Factor3[2]-mean(Factor3)))+mean(Factor3)

###Vector de valores óptimos sin codificar para el factor A, B y C###

V_optimos_nlm<-c(A_onlm_sc,B_onlm_sc,C_onlm_sc)

##########ANALISIS USANSO SIMULACION MONTECARLO##########

Voli<-function()
               { repeat
                      {###Entrada de datos###

                             Intercepto<-240.750
                             Coef_A<--20.417
                             Coef_B<-71.583
                             Coef_C<-17.917
                             Coef_AC<--38.083

                       ###Simulación de valores para A B C AC respectivamente###

                             A<-runif(1,A_onlm-afa,A_onlm+afa)
                             B<-runif(1,B_onlm-afb,B_onlm+afb)
                             C<-runif(1,C_onlm-afc,C_onlm+afc)
                             AC<-A*C

                             Variacion<-rnorm(1,0,8.547)

                      ###Simulación de un valor de Y (Volumen)###

              Y<-Intercepto + Coef_A*A + Coef_B*B + Coef_C*C + Coef_AC*AC + Variacion

                             vol<-Y
                             return(vol)

                      }
               }

Vol<-Voli()

###El siguiente código implementa al programa para que proporcione r replicas de Y
(Volumen)###
```

```
vol<-rep(0,r)

for (i in 1:r)
        {
        vol[i]<-Voli()
        }

###Fracción Defectuosa (fd) y probabilidad de cumplir con las especificaciones (p)###

fc_<-which(vol>=lsl & vol<=usl)
fc<-length(fc_)/r
fd<-1-fc
p<-fc

###Media y desviación estándar del volumen inyectado (mvol y sdvol)

mvol<-mean(vol)
sdvol<-sd(vol)

###RESULTADOS###

V_optimos_nlm<-c(A_onlm_sc,B_onlm_sc,C_onlm_sc)
V_optimos_nlm
fd
p
mvol
sdvol

###Histograma y densidad empírica del volumen inyectado por las unidades simuladas###

hist(vol, breaks=40)
plot(density(vol))
```

## Rutina 12.3.1 Intervalos de confianza para los ICP (Optimización NLM)

```
###DATOS del programa de la simulación Monte Carlo (Optimización NLM)###

datosvol<-vol
tolrange<-usl-lsl

###Datos para el remuestreo bootstrap###

r<-10000
n<-250

ICPi=function()
               {repeat
                       {###Remuestreo de los datos###

                        x<-sample(datosvol,r,TRUE)
                       x2<-sample(x,n,TRUE)
```

```
                        ###Calculo de los cuantiles###

                      LI<-quantile(x,0.00135)
                      MA<-quantile(x,0.5)
                      LS<-quantile(x,0.99865)

                      ###Media y desviación estándar de los datos###

                           mx<-mean(x2)
                          sdx<-sd(x2)

                      ###Construcción del Índice Cp (normal, no-normal)###

                         I_Cp<-(tolrange/(6*(sdx)))
                        I_Cp2<-(tolrange/((LS-LI)))

                      ###Construcción del Índice Cpk (normal, no-normal)###

                              d<-tolrange/2
                              M<-(usl+lsl)/2
                         I_Cpk<-(d-abs(mx-M))/(3*sdx)
                        I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                      ###Construcción del Índice Cpm (normal, no-normal)###

                            T<-M
                        I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                       I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))


      ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                       I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                      I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                      ###Construcción del índice Cs (normal, no-normal)###

                          ite<-n
                 estadistico<-rep(0,ite)
                                  for(i in 1:ite)
                                  {
                                  estadistico[i]<-(x[i]-mx)^3
                                  }
                         mu_3<-mean(estadistico)

                      I_Cs<-(d-abs(mx-M))/(3*sqrt(sdx^2+(mx-T)^2+abs(mu_3/sdx)))
                      I_Cs2<-(d-abs(MA-M))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+abs(mu_3/d)))

                      ###Vector que contiene los valores de los ICP###

          I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                      return(I)

                             }
                      }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###
ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
       {
       indices[i,]<-ICPi()
       }

###Construcción de los Intervalos de Confianza###
```

```
###Estimador bootstrap del Índice Cp (normal, no-normal)###

mediaI_Cp<-mean(indices[,1])
mediaI_Cp2<-mean(indices[,2])

###IC para los índices Cp (normal, no-normal)###

indicesI_Cp<-indices[,1]
IC_I_Cp<-c(quantile(indicesI_Cp,0.025),quantile(indicesI_Cp,0.975))

indicesI_Cp2<-indices[,2]
IC_I_Cp2<-c(quantile(indicesI_Cp2,0.025),quantile(indicesI_Cp2,0.975))

###Estimador bootstrap del Índice Cpk (normal, no-normal)###

mediaI_Cpk<-mean(indices[,3])
mediaI_Cpk2<-mean(indices[,4])

###IC para los índices Cpk (normal, no-normal)###

indicesI_Cpk<-indices[,3]
IC_I_Cpk<-c(quantile(indicesI_Cpk,0.025),quantile(indicesI_Cpk,0.975))

indicesI_Cpk2<-indices[,4]
IC_I_Cpk2<-c(quantile(indicesI_Cpk2,0.025),quantile(indicesI_Cpk2,0.975))

###Estimador bootstrap del Índice Cpm (normal, no-normal)###

mediaI_Cpm<-mean(indices[,5])
mediaI_Cpm2<-mean(indices[,6])

###IC para los índices Cpm (normal, no-normal)###

indicesI_Cpm<-indices[,5]
IC_I_Cpm<-c(quantile(indicesI_Cpm,0.025),quantile(indicesI_Cpm,0.975))

indicesI_Cpm2<-indices[,6]
IC_I_Cpm2<-c(quantile(indicesI_Cpm2,0.025),quantile(indicesI_Cpm2,0.975))

###Estimador bootstrap del Índice Cpmk (normal, no-normal)###

mediaI_Cpmk<-mean(indices[,7])
mediaI_Cpmk2<-mean(indices[,8])

###IC para los índices Cpmk (normal, no-normal)###

indicesI_Cpmk<-indices[,7]
IC_I_Cpmk<-c(quantile(indicesI_Cpmk,0.025),quantile(indicesI_Cpmk,0.975))

indicesI_Cpmk2<-indices[,8]
IC_I_Cpmk2<-c(quantile(indicesI_Cpmk2,0.025),quantile(indicesI_Cpmk2,0.975))

###Estimador bootstrap del Índice Cs (normal, no-normal)###

mediaI_Cs<-mean(indices[,9])
mediaI_Cs2<-mean(indices[,10])

###IC para los indices Cs (normal, no-normal)###

indicesI_Cs<-indices[,9]
IC_I_Cs<-c(quantile(indicesI_Cs,0.025),quantile(indicesI_Cs,0.975))

indicesI_Cs2<-indices[,10]
IC_I_Cs2<-c(quantile(indicesI_Cs2,0.025),quantile(indicesI_Cs2,0.975))


###SALIDA: [Índice de Capacidad, Intervalo de Confianza al 95% (LI, LS)]###

valor<-rep(0,ni*3)
valores<-matrix(valor,ni,3)
```

```
valores[1,1]<-mediaI_Cp
valores[1,2]<-IC_I_Cp[1]
valores[1,3]<-IC_I_Cp[2]

valores[2,1]<-mediaI_Cp2
valores[2,2]<-IC_I_Cp2[1]
valores[2,3]<-IC_I_Cp2[2]

valores[3,1]<-mediaI_Cpk
valores[3,2]<-IC_I_Cpk[1]
valores[3,3]<-IC_I_Cpk[2]

valores[4,1]<-mediaI_Cpk2
valores[4,2]<-IC_I_Cpk2[1]
valores[4,3]<-IC_I_Cpk2[2]

valores[5,1]<-mediaI_Cpm
valores[5,2]<-IC_I_Cpm[1]
valores[5,3]<-IC_I_Cpm[2]

valores[6,1]<-mediaI_Cpm2
valores[6,2]<-IC_I_Cpm2[1]
valores[6,3]<-IC_I_Cpm2[2]

valores[7,1]<-mediaI_Cpmk
valores[7,2]<-IC_I_Cpmk[1]
valores[7,3]<-IC_I_Cpmk[2]

valores[8,1]<-mediaI_Cpmk2
valores[8,2]<-IC_I_Cpmk2[1]
valores[8,3]<-IC_I_Cpmk2[2]

valores[9,1]<-mediaI_Cs
valores[9,2]<-IC_I_Cs[1]
valores[9,3]<-IC_I_Cs[2]

valores[10,1]<-mediaI_Cs2
valores[10,2]<-IC_I_Cs2[1]
valores[10,3]<-IC_I_Cs2[2]

valores
```

## Rutina 12.3.2 Porcentajes de cobertura de los IC (Optimización NLM)

```
##########ANALISIS DE LOS PORCENTAJES DE COBERTURA (Optimización NLM)##########


###Valores de los Intervalos de Confianza Obtenidos con Optimización NLM###


IC_I_Cp
IC_I_Cp2
IC_I_Cpk
IC_I_Cpk2
IC_I_Cpm
IC_I_Cpm2
IC_I_Cpmk
IC_I_Cpmk2
IC_I_Cs
IC_I_Cs2


###DATOS del programa de simulación del volumen (Optimización NLM)###
```

```
datosvol<-volu
r<-10000
n<-250

ICPi=function()
              {repeat
                     {###Remuestreo de los datos###

                       x<-sample(datosvol,r,TRUE)
                     x2<-sample(x,n,TRUE)

                      ###Calculo de los cuantiles###

                     LI<-quantile(x,0.00135)
                     MA<-quantile(x,0.5)
                     LS<-quantile(x,0.99865)

                     ###Media y desviación estándar de los datos###

                          mx<-mean(x2)
                         sdx<-sd(x2)

                     ###Construcción del Índice Cp (normal, no-normal)###

                        I_Cp<-(tolrange/(6*(sdx)))
                       I_Cp2<-(tolrange/(LS-LI))

                     ###Construcción del Índice Cpk (normal, no-normal)###

                            d<-tolrange/2
                            M<-(usl+lsl)/2
                       I_Cpk<-(d-abs(mx-M))/(3*sdx)
                      I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                     ###Construcción del Índice Cpm (normal, no-normal)###

                           T<-M
                       I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                      I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

       ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                      I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                     I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                      ###Construcción del índice Cs (normal, no-normal)###
                     ###Calculo del tercer momento de la variable vol###
```

```
                    mux3<-mean(x2)
                     ite<-n
          estadistico<-rep(0,ite)
                              for(i in 1:ite)
                              {
                              estadistico[i]<-(x2[i]-mux3)^3
                              }
                    mu_3<-mean(estadistico)
        I_Cs<-(d-(abs(mx-M)))/(3*sqrt(sdx^2+(mx-T)^2+(abs(mu_3/sdx))))
        I_Cs2<-(d-(abs(MA-M)))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+(abs(mu_3/d))))

              ###Vector que contiene los valores de los ICP###

           I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

              return(I)

                     }
              }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###

ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
       {
       indices[i,]<-ICPi()
       }

###Calculo del porcentaje de cobertura de los ICP###

pCp<-which(indices[,1]>=IC_I_Cp[1] & indices[,1]<=IC_I_Cp[2])
p_Cp<-length(pCp)/r

pCp2<-which(indices[,2]>=IC_I_Cp2[1] & indices[,2]<=IC_I_Cp2[2])
p_Cp2<-length(pCp2)/r

pCpk<-which(indices[,3]>=IC_I_Cpk[1] & indices[,3]<=IC_I_Cpk[2])
p_Cpk<-length(pCpk)/r

pCpk2<-which(indices[,4]>=IC_I_Cpk2[1] & indices[,4]<=IC_I_Cpk2[2])
p_Cpk2<-length(pCpk2)/r
```

```
pCpm<-which(indices[,5]>=IC_I_Cpm[1] & indices[,5]<=IC_I_Cpm[2])
p_Cpm<-length(pCpm)/r

pCpm2<-which(indices[,6]>=IC_I_Cpm2[1] & indices[,6]<=IC_I_Cpm2[2])
p_Cpm2<-length(pCpm2)/r

pCpmk<-which(indices[,7]>=IC_I_Cpmk[1] & indices[,7]<=IC_I_Cpmk[2])
p_Cpmk<-length(pCpmk)/r

pCpmk2<-which(indices[,8]>=IC_I_Cpmk2[1] & indices[,8]<=IC_I_Cpmk2[2])
p_Cpmk2<-length(pCpmk2)/r

pCs<-which(indices[,9]>=IC_I_Cs[1] & indices[,9]<=IC_I_Cs[2])
p_Cs<-length(pCs)/r

pCs2<-which(indices[,10]>=IC_I_Cs2[1] & indices[,10]<=IC_I_Cs2[2])
p_Cs2<-length(pCs2)/r

###Porcentajes de cobertura###

pc<-c(p_Cp,p_Cpk,p_Cpm,p_Cpmk,p_Cs,p_Cp2,p_Cpk2,p_Cpm2,p_Cpmk2,p_Cs2)

PC<-matrix(pc,ni/2,2)

PC
```

## Rutina 12.4 Optimización Taguchi

```
##########Optimizacion Taguchi##########

###Transformación de los datos a SN (signal-ratios)###

renglones<-8
columnas<-3
datos<-
c(126,141,122,183,168,164,284,283,275,300,318,310,249,242,242,125,128,140,387,392,391,284,26
9,255)
Datos<-matrix(datos,renglones,columnas,TRUE)

sm<-rep(0,renglones)
datosr<-rep(0,columnas)
vm<-rep(0,renglones)
sn<-rep(0,renglones)
for(i in 1:renglones)
{
n<-columnas
datosr<-Datos[i,]
sm[i]<-(sum(datosr))^2/n
y_2<-rep(0,n)
for(j in 1:n)
{y_2[j]<-(datosr[j]^2)}
sy_2<-sum(y_2)
vm[i]<-(sy_2-sm[i])/(n-1)
sn[i]<-10*log10((sm[i]-vm[i])/(n*vm[i]))
}
SN<-t(t(sn))
```

```
###ANOVA para identificar los factores que afectan la variabilidad del proceso###

renglonesAV<-renglones
columnasAV<-7
numceldasav<-renglonesAV*columnasAV
datosav<-c(sn[1],1,1,1,1,1,1,sn[2],2,1,1,2,2,1,sn[3],1,2,1,2,1,2,sn[4],2,2,1,1,2,2,
           sn[5],1,1,2,1,2,2,sn[6],2,1,2,2,1,2,sn[7],1,2,2,2,2,1,sn[8],2,2,2,1,1,1)
DatosAV<-matrix(datosav,renglonesAV,columnasAV,TRUE)
DatosSNV<-as.data.frame(DatosAV)
LinearModel.1 <- lm(V1  ~ V2  + V3  + V4 + V5 + V6 + V7 , data=DatosSNV)
anova(LinearModel.1)

###Observando la tabla de ANOVA los factores A, B y C son signicicativos. Las interacciones
AB, AC y BC también son significativas por tanto afectan la variabilidad del proceso.
Entonces se elegirán los niveles de cada factor que tenga mayor promedio###

###Calculo de la media de A1###

AV__1<-min(DatosAV[,2])
AV_1<-which(DatosAV[,2]==AV__1)
AV1<-DatosAV[c(AV_1),]
mediaAV1<-mean(AV1[,1])

###Calculo de la media de A2###

AV__2<-max(DatosAV[,2])
AV_2<-which(DatosAV[,2]==AV__2)
AV2<-DatosAV[c(AV_2),]
mediaAV2<-mean(AV2[,1])

###Vector de medias para cada una de los niveles del factor A###

av1av2<-c(mediaAV1,mediaAV2)
AV1AV2<-t(av1av2)
AV_opt<-max(AV1AV2[1,])
AV_optimo<-which(AV1AV2[1,]==AV_opt)

###Calculo de la media de B1###

BV__1<-min(DatosAV[,3])
BV_1<-which(DatosAV[,3]==BV__1)
BV1<-DatosAV[c(BV_1),]
mediaBV1<-mean(BV1[,1])

###Calculo de la media de B2###

BV__2<-max(DatosAV[,3])
BV_2<-which(DatosAV[,3]==BV__2)
BV2<-DatosAV[c(BV_2),]
mediaBV2<-mean(BV2[,1])

###Vector de medias para cada uno de los niveles del factor B###

bv1bv2<-c(mediaBV1,mediaBV2)
BV1BV2<-t(bv1bv2)
BV_opt<-max(BV1BV2[1,])
BV_optimo<-which(BV1BV2[1,]==BV_opt)

###Calculo de la media de C1###

CV__1<-min(DatosAV[,4])
CV_1<-which(DatosAV[,4]==CV__1)
CV1<-DatosAV[c(CV_1),]
mediaCV1<-mean(CV1[,1])

###Calculo de la media de C2###

CV__2<-max(DatosAV[,4])
CV_2<-which(DatosAV[,4]==CV__2)
CV2<-DatosAV[c(CV_2),]
mediaCV2<-mean(CV2[,1])
```

```
###Vector de medias para cada uno de los niveles del factor C###

cv1cv2<-c(mediaCV1,mediaCV2)
CV1CV2<-t(cv1cv2)
CV_opt<-max(CV1CV2[1,])
CV_optimo<-which(CV1CV2[1,]==CV_opt)

###Calculo de la media de A1*B1###

ABV__11<-min(AV1[,3])
ABV_11<-which(AV1[,3]==ABV__11)
ABV11<-AV1[c(ABV_11),]
mediaABV11<-mean(ABV11[,1])

###Calculo de la media de A1*B2###

ABV__12<-max(AV1[,3])
ABV_12<-which(AV1[,3]==ABV__12)
ABV12<-AV1[c(ABV_12),]
mediaABV12<-mean(ABV12[,1])

###Calculo de la media de A2*B1###

ABV__21<-min(AV2[,3])
ABV_21<-which(AV2[,3]==ABV__21)
ABV21<-AV2[c(ABV_21),]
mediaABV21<-mean(ABV21[,1])

###Calculo de la media de A2*B2###

ABV__22<-max(AV2[,3])
ABV_22<-which(AV2[,3]==ABV__22)
ABV22<-AV2[c(ABV_22),]
mediaABV22<-mean(ABV22[,1])

###Vector de medias (A1*B1,A1*B2,A2*B1,A2*B2)###

mediaAB<-c(mediaABV11,mediaABV12,mediaABV21,mediaABV22)
ABV_opt<-max(mediaAB)

###De este vector se observa que para la interacción A1B2 maximiza la media de los SN###

###Calculo de la media de B1*C1###

BCV__11<-min(BV1[,4])
BCV_11<-which(BV1[,4]==BCV__11)
BCV11<-BV1[c(BCV_11),]
mediaBCV11<-mean(BCV11[,1])

###Calculo de la media de B1*C2###

BCV__12<-max(BV1[,4])
BCV_12<-which(BV1[,4]==BCV__12)
BCV12<-BV1[c(BCV_12),]
mediaBCV12<-mean(BCV12[,1])

###Calculo de la media de B2*C1###

BCV__21<-min(BV2[,4])
BCV_21<-which(BV2[,4]==BCV__21)
BCV21<-BV2[c(BCV_21),]
mediaBCV21<-mean(BCV21[,1])

###Calculo de la media de B2*C2###

BCV__22<-max(BV2[,4])
BCV_22<-which(BV2[,4]==BCV__22)
BCV22<-BV2[c(BCV_22),]
mediaBCV22<-mean(BCV22[,1])
```

```
###Vector de medias (B1*C1,B1*C2,B2*C1,B2*C2)###

mediaBC<-c(mediaBCV11,mediaBCV12,mediaBCV21,mediaBCV22)
BCV_opt<-max(mediaBC)

###De este vector se observa que para la interacción B2C2 maximiza la media de los SN###

###Calculo de la media de A1*C1###

ACV__11<-min(AV1[,4])
ACV_11<-which(AV1[,4]==ACV__11)
ACV11<-AV1[c(ACV_11),]
mediaACV11<-mean(ACV11[,1])

###Calculo de la media de A1*C2###

ACV__12<-max(AV1[,4])
ACV_12<-which(AV1[,4]==ACV__12)
ACV12<-AV1[c(ACV_12),]
mediaACV12<-mean(ACV12[,1])

###Calculo de la media de A2*C1###

ACV__21<-min(AV2[,4])
ACV_21<-which(AV2[,4]==ACV__21)
ACV21<-AV2[c(ACV_21),]
mediaACV21<-mean(ACV21[,1])

###Calculo de la media de A2*C2###

ACV__22<-max(AV2[,4])
ACV_22<-which(AV2[,4]==ACV__22)
ACV22<-AV2[c(ACV_22),]
mediaACV22<-mean(ACV22[,1])

#Vector de medias (A1*C1,A1*C2,A2*C1,A2*C2)###

mediaAC<-c(mediaACV11,mediaACV12,mediaACV21,mediaACV22)
ACV_opt<-max(mediaAC)

###Promedios de los niveles óptimos para cada uno de los factores###

renglonesAVM<-renglones
columnasAVM<-7
numceldasavm<-renglonesAVM*columnasAVM
datosavm<-
c(mean(Datos[1,]),1,1,1,1,1,1,mean(Datos[2,]),2,1,1,2,2,1,mean(Datos[3,]),1,2,1,2,1,2,mean(D
atos[4,]),2,2,1,1,2,2,mean(Datos[5,]),1,1,2,1,2,2,mean(Datos[6,]),2,1,2,2,1,2,mean(Datos[7,]
),1,2,2,2,2,1,mean(Datos[8,]),2,2,2,1,1,1)
DatosAVM<-matrix(datosavm,renglonesAVM,columnasAVM,TRUE)

###Calculo de la media de A1###

AVM__1<-min(DatosAV[,2])
AVM_1<-which(DatosAVM[,2]==AVM__1)
AVM1<-DatosAVM[c(AVM_1),]
mediaAVM1<-mean(AVM1[,1])
AVM_opt<-mediaAVM1

###Calculo de la media de B2###

BVM__2<-max(DatosAVM[,3])
BVM_2<-which(DatosAVM[,3]==BVM__2)
BVM2<-DatosAVM[c(BVM_2),]
mediaBVM2<-mean(BVM2[,1])
BVM_opt<-mediaBVM2

###Calculo de la media de C2###

CVM__2<-max(DatosAVM[,4])
CVM_2<-which(DatosAVM[,4]==CVM__2)
```

```
CVM2<-DatosAVM[c(CVM_2),]
mediaCVM2<-mean(CVM2[,1])
CVM_opt<-mediaCVM2

###Calculo de la media de A1*B2###

ABVM__12<-max(AVM1[,3])
ABVM_12<-which(AVM1[,3]==ABVM__12)
ABVM12<-AVM1[c(ABVM_12),]
mediaABVM12<-mean(ABVM12[,1])
ABVM_opt<-mediaABVM12

###Calculo de la media de B2*C2###

BCVM__22<-max(BVM2[,4])
BCVM_22<-which(BVM2[,4]==BCVM__22)
BCVM22<-BVM2[c(BCVM_22),]
mediaBCVM22<-mean(BCVM22[,1])
BCVM_opt<-mediaBCVM22

###Calculo de la media de A1*C2###

ACVM__12<-max(AVM1[,4])
ACVM_12<-which(AVM1[,4]==ACVM__12)
ACVM12<-AVM1[c(ACVM_12),]
mediaACVM12<-mean(ACVM12[,1])
ACVM_opt<-mediaACVM12

###De este vector se observa que para la interacción A1C2 maximiza la media de los SN###

###Análisis Grafico###

###GRAFICOS###
x<-c(1,2)

###Gráficos señal a ruido###

AV<-c(mediaAV1,mediaAV2)
BV<-c(mediaBV1,mediaBV2)
CV<-c(mediaCV1,mediaCV2)

###Factor A###

matplot(x,AV,xaxt="n","l",main="Señal a Ruido",xlab="NIVEL",ylab="Factor A")
axis(1,at=c(1,2),labels=expression(1,2))

###Factor B###

matplot(x,BV,xaxt="n","l",main="Señal a Ruido",xlab="NIVEL",ylab="Factor B")
axis(1,at=c(1,2),labels=expression(1,2))

###Factor C###

matplot(x,CV,xaxt="n","l",main="Señal a Ruido",xlab="NIVEL",ylab="Factor C")
axis(1,at=c(1,2),labels=expression(1,2))

####RESULTADOS####

anova(LinearModel.1)

###Y OPTIMO###

Ypromedio<-mean(datos)
Yopt<-Ypromedio+(AVM_opt-Ypromedio)+(BVM_opt-Ypromedio)+(CVM_opt-Ypromedio)

###Valores omitidos por el sesgo que causan en el Y optimo respecto al valor meta###

###+(ABVM_opt-Ypromedio)+(BCVM_opt-Ypromedio)+(ACVM_opt-Ypromedio)###

Yopt
```

## Rutina 12.4.1 Simulación Monte Carlo (Optimización Taguchi)

```
##########ANALISIS USANSO SIMULACION MONTECARLO##########

###Simulación Monte Carlo (Modelo bajo optimización Taguchi)###

###Entrada de datos(número de replicas y límites de especificación)###

r<-100000
usl<-330
lsl<-270

###Niveles de los Factores###

Factor1<-c(500,900)
Factor2<-c(6,9)
Factor3<-c(0.3,0.6)

###Valores del diseño bajo la optimización Taguchi###

Fac_A<-c(450,550)
Fac_B<-c(8.85,9.15)
Fac_C<-c(0.57,0.63)

###Codificación de los valores del diseño###

F_A<-c((Fac_A[1]-mean(Factor1))/(mean(Factor1)-Factor1[1]),(Fac_A[2]-
mean(Factor1))/(mean(Factor1)-Factor1[1]))
F_B<-c((Fac_B[1]-mean(Factor2))/(mean(Factor2)-Factor2[1]),(Fac_B[2]-
mean(Factor2))/(mean(Factor2)-Factor2[1]))
F_C<-c((Fac_C[1]-mean(Factor3))/(mean(Factor3)-Factor3[1]),(Fac_C[2]-
mean(Factor3))/(mean(Factor3)-Factor3[1]))

###Amplitudes codificadas de cada uno de los factores del diseño###

AFAC<-F_A[2]-F_A[1]
AFBC<-F_B[2]-F_B[1]
AFCC<-F_C[2]-F_C[1]

###Simulación de un inyector###

Voli<-function()
              { repeat
                     {###Entrada de datos del modelo inicial###

                            Intercepto<-240.750
                            Coef_A<--20.417
                            Coef_B<-71.583
```

```
                        Coef_C<-17.917
                        Coef_AC<--38.083

                    ###Simulación de valores para A B C AC respectivamente###

                        A<-runif(1,F_A[1],F_A[2])
                        B<-runif(1,F_B[1],F_B[2])
                        C<-runif(1,F_C[1],F_C[2])
                        AC<-A*C

                        Variacion<-rnorm(1,0,8.547)

                    ###Simulación de un valor de Y (Volumen)###

    Y<-Intercepto + Coef_A*A + Coef_B*B + Coef_C*C + Coef_AC*AC + Variacion

                        vol<-Y
                        return(vol)

                    }
            }

Vol<-Voli()

###El siguiente código proporciona r repeticiones de Y (Volumen)###

vol<-rep(0,r)

for (i in 1:r)
    {
    vol[i]<-Voli()
    }

###Calculo de la Fracción Defectuosa (fd)###

fc_<-which(vol>=lsl & vol<=usl)
fc<-length(fc_)/r
fd<-1-fc
fd

###Media y desviación estándar del volumen inyectado (mvol y sdvol)###

mvol<-mean(vol)
sdvol<-sd(vol)
mvol
sdvol

###Histograma y densidad empírica del volumen inyectado por las unidades simuladas###
```

```
hist(vol, breaks=40)
plot(density(vol))
```

## Rutina 12.4.2 Intervalos de confianza para los ICP (Optimización Taguchi)

```
###DATOS del programa de la simulación Monte Carlo (Optimización Taguchi)###

datosvol<-vol
tolrange<-usl-lsl

###Datos para el remuestreo bootstrap###

r<-10000
n<-250

ICPi=function()
               {repeat
                      {###Remuestreo de los datos###

                       x<-sample(datosvol,r,TRUE)
                      x2<-sample(x,n,TRUE)

                       ###Cálculo de los cuantiles###

                      LI<-quantile(x,0.00135)
                      MA<-quantile(x,0.5)
                      LS<-quantile(x,0.99865)

                      ###Media y desviación estándar de los datos###

                           mx<-mean(x2)
                          sdx<-sd(x2)

                      ###Construcción del Índice Cp (normal, no-normal)###

                         I_Cp<-(tolrange/(6*(sdx)))
                        I_Cp2<-(tolrange/((LS-LI)))

                      ###Construcción del Índice Cpk (normal, no-normal)###

                             d<-tolrange/2
                             M<-(usl+lsl)/2
                         I_Cpk<-(d-abs(mx-M))/(3*sdx)
                        I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                      ###Construcción del Índice Cpm (normal, no-normal)###

                           T<-M
                        I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                       I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))


      ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                       I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                      I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                      ###Construcción del índice Cs (normal, no-normal)###

                          ite<-n
                  estadistico<-rep(0,ite)
                                for(i in 1:ite)
                                {
                                estadistico[i]<-(x[i]-mx)^3
                                }
                        mu_3<-mean(estadistico)
```

```
                        I_Cs<-(d-abs(mx-M))/(3*sqrt(sdx^2+(mx-T)^2+abs(mu_3/sdx)))
                        I_Cs2<-(d-abs(MA-M))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+abs(mu_3/d)))

                        ###Vector que contiene los valores de los ICP###

            I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                        return(I)

                                }
                        }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###
ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)

for (i in 1:r)
        {
        indices[i,]<-ICPi()
        }

###Construcción de los Intervalos de Confianza###

###Estimador bootstrap del Índice Cp (normal, no-normal)###

mediaI_Cp<-mean(indices[,1])
mediaI_Cp2<-mean(indices[,2])

###IC para los índices Cp (normal, no-normal)###

indicesI_Cp<-indices[,1]
IC_I_Cp<-c(quantile(indicesI_Cp,0.025),quantile(indicesI_Cp,0.975))

indicesI_Cp2<-indices[,2]
IC_I_Cp2<-c(quantile(indicesI_Cp2,0.025),quantile(indicesI_Cp2,0.975))

###Estimador bootstrap del Índice Cpk (normal, no-normal)###

mediaI_Cpk<-mean(indices[,3])
mediaI_Cpk2<-mean(indices[,4])

###IC para los índices Cpk (normal, no-normal)###

indicesI_Cpk<-indices[,3]
IC_I_Cpk<-c(quantile(indicesI_Cpk,0.025),quantile(indicesI_Cpk,0.975))

indicesI_Cpk2<-indices[,4]
IC_I_Cpk2<-c(quantile(indicesI_Cpk2,0.025),quantile(indicesI_Cpk2,0.975))

###Estimador bootstrap del Índice Cpm (normal, no-normal)###

mediaI_Cpm<-mean(indices[,5])
mediaI_Cpm2<-mean(indices[,6])

###IC para los índices Cpm (normal, no-normal)###

indicesI_Cpm<-indices[,5]
IC_I_Cpm<-c(quantile(indicesI_Cpm,0.025),quantile(indicesI_Cpm,0.975))

indicesI_Cpm2<-indices[,6]
IC_I_Cpm2<-c(quantile(indicesI_Cpm2,0.025),quantile(indicesI_Cpm2,0.975))

###Estimador bootstrap del Índice Cpmk (normal, no-normal)###

mediaI_Cpmk<-mean(indices[,7])
mediaI_Cpmk2<-mean(indices[,8])
```

```
###IC para los índices Cpmk (normal, no-normal)###

indicesI_Cpmk<-indices[,7]
IC_I_Cpmk<-c(quantile(indicesI_Cpmk,0.025),quantile(indicesI_Cpmk,0.975))

indicesI_Cpmk2<-indices[,8]
IC_I_Cpmk2<-c(quantile(indicesI_Cpmk2,0.025),quantile(indicesI_Cpmk2,0.975))

###Estimador bootstrap del Índice Cs (normal, no-normal)###

mediaI_Cs<-mean(indices[,9])
mediaI_Cs2<-mean(indices[,10])

###IC para los indices Cs (normal, no-normal)###

indicesI_Cs<-indices[,9]
IC_I_Cs<-c(quantile(indicesI_Cs,0.025),quantile(indicesI_Cs,0.975))

indicesI_Cs2<-indices[,10]
IC_I_Cs2<-c(quantile(indicesI_Cs2,0.025),quantile(indicesI_Cs2,0.975))


###SALIDA: [Índice de Capacidad, Intervalo de Confianza al 95% (LI, LS)]###

valor<-rep(0,ni*3)
valores<-matrix(valor,ni,3)

valores[1,1]<-mediaI_Cp
valores[1,2]<-IC_I_Cp[1]
valores[1,3]<-IC_I_Cp[2]

valores[2,1]<-mediaI_Cp2
valores[2,2]<-IC_I_Cp2[1]
valores[2,3]<-IC_I_Cp2[2]

valores[3,1]<-mediaI_Cpk
valores[3,2]<-IC_I_Cpk[1]
valores[3,3]<-IC_I_Cpk[2]

valores[4,1]<-mediaI_Cpk2
valores[4,2]<-IC_I_Cpk2[1]
valores[4,3]<-IC_I_Cpk2[2]

valores[5,1]<-mediaI_Cpm
valores[5,2]<-IC_I_Cpm[1]
valores[5,3]<-IC_I_Cpm[2]

valores[6,1]<-mediaI_Cpm2
valores[6,2]<-IC_I_Cpm2[1]
valores[6,3]<-IC_I_Cpm2[2]

valores[7,1]<-mediaI_Cpmk
valores[7,2]<-IC_I_Cpmk[1]
valores[7,3]<-IC_I_Cpmk[2]

valores[8,1]<-mediaI_Cpmk2
valores[8,2]<-IC_I_Cpmk2[1]
valores[8,3]<-IC_I_Cpmk2[2]

valores[9,1]<-mediaI_Cs
valores[9,2]<-IC_I_Cs[1]
valores[9,3]<-IC_I_Cs[2]

valores[10,1]<-mediaI_Cs2
valores[10,2]<-IC_I_Cs2[1]
valores[10,3]<-IC_I_Cs2[2]

valores
```

## Rutina 12.4.3 Porcentajes de cobertura de los IC (Optimización Taguchi)

```
##########ANALISIS DE LOS PORCENTAJES DE COBERTURA (Optimización Taguchi)##########

###Valores de los Intervalos de Confianza Obtenidos con Optimización Taguchi###

IC_I_Cp
IC_I_Cp2
IC_I_Cpk
IC_I_Cpk2
IC_I_Cpm
IC_I_Cpm2
IC_I_Cpmk
IC_I_Cpmk2
IC_I_Cs
IC_I_Cs2

###DATOS del programa de simulación del volumen (Optimización NLM)###

datosvol<-volu
r<-10000
n<-250

ICPi=function()
               {repeat
                       {###Remuestreo de los datos###

                         x<-sample(datosvol,r,TRUE)
                       x2<-sample(x,n,TRUE)

                        ###Calculo de los cuantiles###

                       LI<-quantile(x,0.00135)
                       MA<-quantile(x,0.5)
                       LS<-quantile(x,0.99865)

                       ###Media y desviación estándar de los datos###

                            mx<-mean(x2)
                           sdx<-sd(x2)

                       ###Construcción del Índice Cp (normal, no-normal)###

                          I_Cp<-(tolrange/(6*(sdx)))
                         I_Cp2<-(tolrange/(LS-LI))

                       ###Construcción del Índice Cpk (normal, no-normal)###
```

```
                    d<-tolrange/2
                    M<-(usl+lsl)/2
                  I_Cpk<-(d-abs(mx-M))/(3*sdx)
                 I_Cpk2<-(d-abs(MA-M))/((LS-LI)/2)

                ###Construcción del Índice Cpm (normal, no-normal)###

                    T<-M
                  I_Cpm<-d/(3*(sqrt((sdx^2)+(mx-T)^2)))
                 I_Cpm2<-d/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

        ###Construcción del índice Cpmk conocido como I de Taguchi (normal, no-normal)###

                 I_Cpmk<-(d-abs(mx-M))/(3*(sqrt((sdx^2)+(mx-T)^2)))
                I_Cpmk2<-(d-abs(MA-M))/(3*(sqrt((((LS-LI)/6)^2)+(MA-T)^2)))

                ###Construcción del índice Cs (normal, no-normal)###
               ###Calculo del tercer momento de la variable vol###

                   mux3<-mean(x2)
                    ite<-n
          estadistico<-rep(0,ite)
                              for(i in 1:ite)
                              {
                              estadistico[i]<-(x2[i]-mux3)^3
                              }
                   mu_3<-mean(estadistico)
        I_Cs<-(d-(abs(mx-M)))/(3*sqrt(sdx^2+(mx-T)^2+(abs(mu_3/sdx))))
        I_Cs2<-(d-(abs(MA-M)))/(3*sqrt((((LS-LI)/6)^2)+(MA-T)^2+(abs(mu_3/d))))

                ###Vector que contiene los valores de los ICP###

            I<-c(I_Cp,I_Cp2,I_Cpk,I_Cpk2,I_Cpm,I_Cpm2,I_Cpmk,I_Cpmk2,I_Cs,I_Cs2)

                return(I)

                       }
                }

ICP<-ICPi()

###Lo siguiente implementa al programa para que nos proporcione r replicas de ICP###
###ni= numero de índices###

ni<-10
indices_<-rep(0,ni*r)
indices<-matrix(indices_,r,ni)
```

```
for (i in 1:r)
        {
        indices[i,]<-ICPi()
        }

###Calculo del porcentaje de cobertura de los ICP###

pCp<-which(indices[,1]>=IC_I_Cp[1] & indices[,1]<=IC_I_Cp[2])
p_Cp<-length(pCp)/r

pCp2<-which(indices[,2]>=IC_I_Cp2[1] & indices[,2]<=IC_I_Cp2[2])
p_Cp2<-length(pCp2)/r

pCpk<-which(indices[,3]>=IC_I_Cpk[1] & indices[,3]<=IC_I_Cpk[2])
p_Cpk<-length(pCpk)/r

pCpk2<-which(indices[,4]>=IC_I_Cpk2[1] & indices[,4]<=IC_I_Cpk2[2])
p_Cpk2<-length(pCpk2)/r

pCpm<-which(indices[,5]>=IC_I_Cpm[1] & indices[,5]<=IC_I_Cpm[2])
p_Cpm<-length(pCpm)/r

pCpm2<-which(indices[,6]>=IC_I_Cpm2[1] & indices[,6]<=IC_I_Cpm2[2])
p_Cpm2<-length(pCpm2)/r

pCpmk<-which(indices[,7]>=IC_I_Cpmk[1] & indices[,7]<=IC_I_Cpmk[2])
p_Cpmk<-length(pCpmk)/r

pCpmk2<-which(indices[,8]>=IC_I_Cpmk2[1] & indices[,8]<=IC_I_Cpmk2[2])
p_Cpmk2<-length(pCpmk2)/r

pCs<-which(indices[,9]>=IC_I_Cs[1] & indices[,9]<=IC_I_Cs[2])
p_Cs<-length(pCs)/r

pCs2<-which(indices[,10]>=IC_I_Cs2[1] & indices[,10]<=IC_I_Cs2[2])
p_Cs2<-length(pCs2)/r

###Porcentajes de cobertura###

pc<-c(p_Cp,p_Cpk,p_Cpm,p_Cpmk,p_Cs,p_Cp2,p_Cpk2,p_Cpm2,p_Cpmk2,p_Cs2)

PC<-matrix(pc,ni/2,2)

PC
```

## 13. BIBLIOGRAFIA:


1. Alonso, A. M. 2002. Un ejemplo de bootstrap suavizado. Lecturas matemáticas. 23: 11-24.
2. Aspray, W. 1993. John von Neumann y los orígenes de la computación moderna. Gedisa. Barcelona, España. 1993. pp: 3-45.
3. Banks, J. 1998. Handbook of simulation: Principles, Methodology, Advances, Applications, and Practice. John Wiley & Sons. New York. 849 p.
4. Chen, K. S., W. L. Pearn, and P. C. Lin. 2003. Capability measures for processes with multiple characteristics. Quality & Reliability Engineering International. 19: 101-110.
5. Choi, B. C., and D. B. Owen. 1990. A study of a New Process Capability Index. Communications in Statistics – Theory and Methods. 19: 1231-1245.
6. Clemen, R. T., and R. L. Winkler. 1999. Combining Probability Distributions From Experts in Risk Analysis. Risk Analysis. 19: 187-203.
7. Clemen, R. T., and T. Reilly. 1999. Making Hard Decisions with Decision Tools. Duxbury Press. Belmont, CA. pp: 1-43.
8. Clements, J. A. 1989. Process Capability Calculations for Non-normal Distributions. Quality Progress. 22: 95-100.
9. Collins, A. J. 1995. Bootstrap Confidence Limits On Process Capability Indices. Statistician. 44: 373-378.
10. Efron, B. 1979a. Bootstrap Methods: Another look at the Jackknife. Annals of Statistics. 7: 1-26.
11. Efron, B. 1979b. Computers and the theory of statistics: thinking the unthinkable. Siam Review. 21: 460-480.
12. Efron, B., and R. J. Tibshirani. 1993. An introduction to the bootstrap. Chapman & Hall. New York. 438 p.
13. Guevara, R. D., y J. A. Vargas. 2006. Intervalos de confianza para los índices de capacidad Cpm y Cpmk en procesos estacionarios gaussianos. Revista Colombiana de Estadística. 29: 153-162.

14. Hsiang, T. C., and G. Taguchi. 1985. Tutorial on Quality Control and Assurance - The Taguchi Methods. Joint Meetings of the American Statistical Association. Las Vegas, Nevada. 188 p.
15. Hunter, J. 1987. Signal to Noise Ratio Debated. Quality Progress. 20: 7-9.
16. Judge, G. 1999. Simple Monte Carlo studies on a spreadsheet. Computers in Higher Education Economics Review. 13: 12-14.
17. Juran, J. M. 1974. Quality Control Handbook. 3rd edition. McGraw-Hill. New York. 1509 p.
18. Kane, V. E. 1986. Process capability indices. Journal of Quality Technology. 18: 41–52.
19. Kiefer, J., and J. Wolfowitz. 1952. Stochastic Estimation of the Maximum of Regresion Function. Ann. Math. Stat. 23: 462-466.
20. Kotz, S., and N. L. Johnson. 2002. Process Capability Indices - A Review, 1992 – 2000 with discussion. Journal of Quality Technology. 34: 2-53.
21. Law, A., and D. Kelton. 2004. Simulation Modeling and Analysis. 3rd edition. McGraw-Hill. New York. pp 84-85
22. Montgomery, D. C. 1991. Introduction to Statistical Quality Control. John Wiley & Son. New York. 797 p.
23. Pearn, W., P. Lin, and K. Chen. 2002. Estimating Process Capability Index Cpmk for Asymmetric Tolerances: Distributional Properties. Metrika. 54: 261-279.
24. Robbins, H. S., and S. Monro. 1951. Stochastic Approximation Methods. Ann. Math. Stat. 22: 400-407.
25. Roy, R. 1990. A Primer on the Taguchi Method. Van Nostrand Reinhold. New York. 247 p.
26. Sleeper, A. 2005. Design for Six Sigma Statistics: 59 Tools for Diagnosing and Solving Problems in DFSS Initiatives. McGraw-Hill. New York. 854 p.
27. Slovic, P., M. L. Finucane, E. Peters, and D. G. MacGregor. 2004. Risk as analysis and risk as feelings: Some thoughts about affect, reason, risk, and rationality. Risk Analysis. 24: 311-322.
28. Solanas, A., y V. Sierra. 1992. Bootstrap: Fundamentos e Introducción a sus Aplicaciones. Anuario de Psicología. 55: 143-154.

29. Sullivan, L. P. 1987. The Power of Taguchi Methods, Quality Progress. 20: 76-79.
30. Taguchi, G., A. Elsayed, and T. C. Hsiang. 1989. Quality Engineering in Production Systems. McGraw-Hill. New York. 173 p.
31. Tjalling, J. Y. 1995. Historical development of the Newton-Raphson method. Siam Review. 37: 531–551.
32. Wright, P. A. 1995. A Process Capability Index Sensitive to Skewness. Journal of Statistical Computation and Simulation. 52: 195-203.