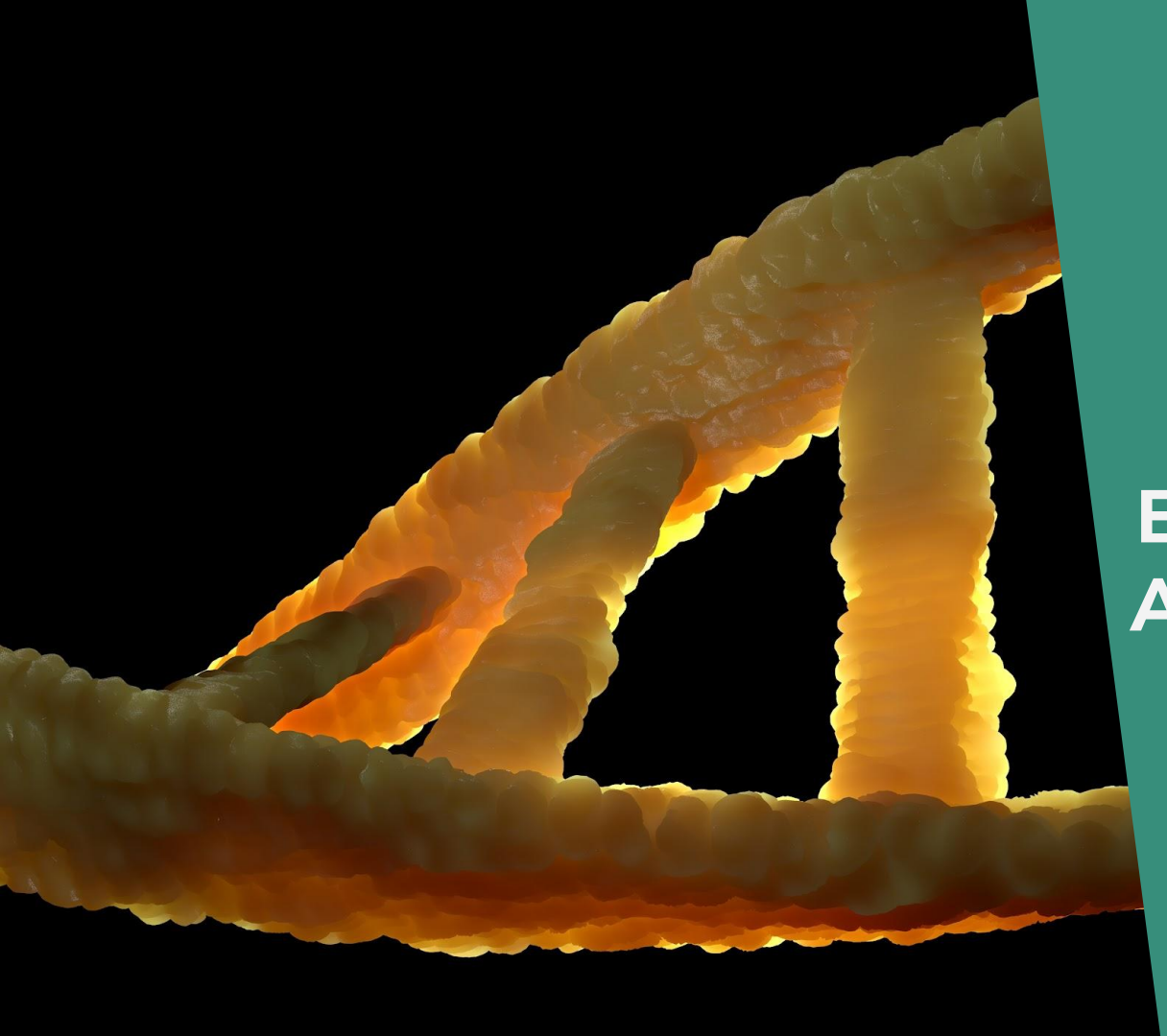


Optimización del Método de Suavizamiento Exponencial con un Algoritmo Genético

GRUPO 1

01

INTRODUCCIÓN

ALGORITMO GENÉTICO

Es una técnica de optimización y búsqueda basada en los principios de genética y selección natural. Un GA permite que una población compuesta por muchos individuos evolucione bajo reglas de selección específicas hacia un estado que maximice su aptitud.

Terminología alineada con Algoritmo Genético

Población

Subconjunto de todas las posibles soluciones. (alpha)

Función Objetivo

Mejora una salida mediante el uso de una entrada específica.

Cromosoma

Soluciones de la población.

Selección

Recoger los individuos más aptos de una generación.

Cruce

Crea 2 nuevos individuos, basados en los genes de dos soluciones.

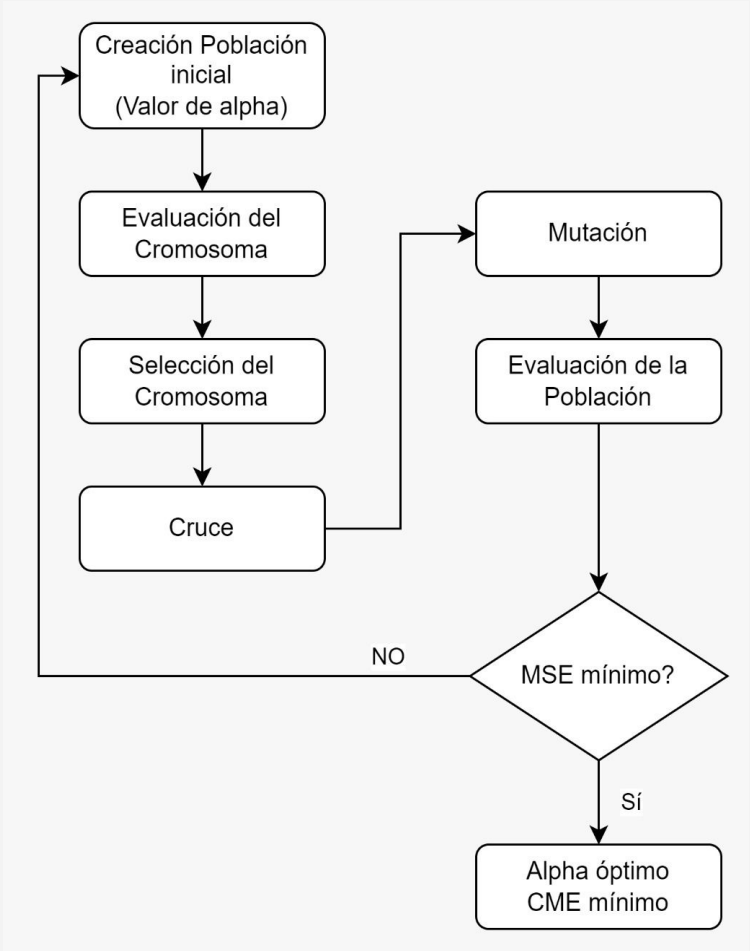
Pr: 0.6 a 0.9

Mutación

Cambiar un gen al azar en el individuo.

Pr: 0.1

DIAGRAMA DE FLUJO



02
TRATAMIENTO DE DATOS

DATASET

DATASET	DESCRIPCIÓN	TEMA	CANTIDAD DE INSTANCIAS
1	**Behavior of the urban traffic of the city of Sao Paulo in Brazil**	Tráfico Urbano	135
2	**Hungarian Chickenpox Cases Data Set**	Medicina	521
3	**Air Quality Data Set**	Ingeniería Ambiental	9358

PASO 1
Datos NA

- Se cargaron los datasets y se verificaron si existían datos faltantes. Siendo este el caso, se procedió a Reemplazar por la media de las variables en los datos faltantes para los 3 datasets.
- Entonces realizando un código que indique si existen datos faltantes en los dataset.

```r
#Función para valores faltantes (NA)
datos_na <- function(x){
  if (sum(is.na(x))>0){
    x[is.na(x)] = mean(x)
    print("**Datos Faltantes (NA) reemplazados**")
  }else{
    print("**No tiene valores faltantes (NA)**")
  }
}

#Dataset 1
for( i in colnames(data1)){
  print(paste("Nombre Columna: ",colnames(data1[i])))
  datos_na(data1[[i]])
}
#Dataset 2
for( i in colnames(data2)){
  print(paste("Nombre Columna: ",colnames(data2[i])))
  datos_na(data2[[i]])
}
#Dataset 3
for( i in colnames(data3)){
  print(paste("Nombre Columna: ",colnames(data3[i])))
  datos_na(data3[[i]])
}
```

Reemplazo de datos faltantes

Código

PASO 2
Variables con Distribución Normal

- Se realiza una función para evaluar la normalidad de todas las columnas de los 3 datasets mediante la prueba de *Kolmogorov Smirnov* (número de datos >50).

p-value > alpha (nivel de significancia) → tiene distribución normal

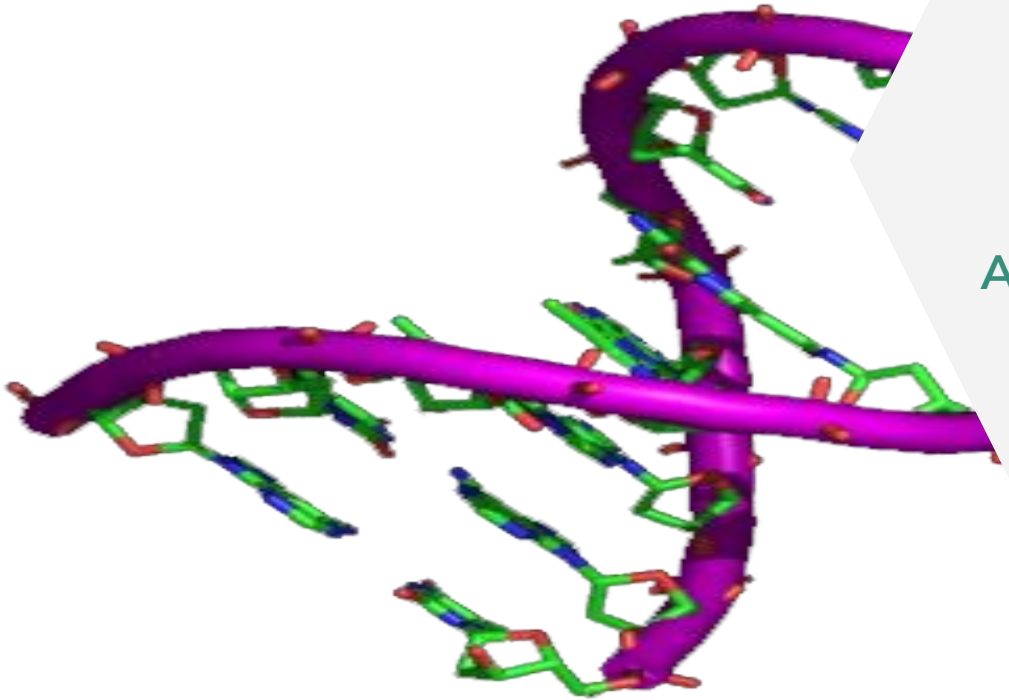
Variable con mejor p-value

Código

```
var_normal <- function(x){
  kolg_srmi <- cbind()
  for( i in colnames(x)){
    k_test <- lillie.test(x[[i]])$p.value
    kolg_srmi <- cbind(kolg_srmi,c(k_test))
  }
  max_Kt=kolg_srmi[which.max(kolg_srmi)]
  indMax_Kt = which.max(kolg_srmi)
  column_nom <- colnames(x[indMax_Kt])
  print(paste("p-value max = ",max_Kt))
  print(paste("Índice de columna: ",indMax_Kt))
  print(paste("Nombre de columna: ",column_nom))
}
```

RESULTADO

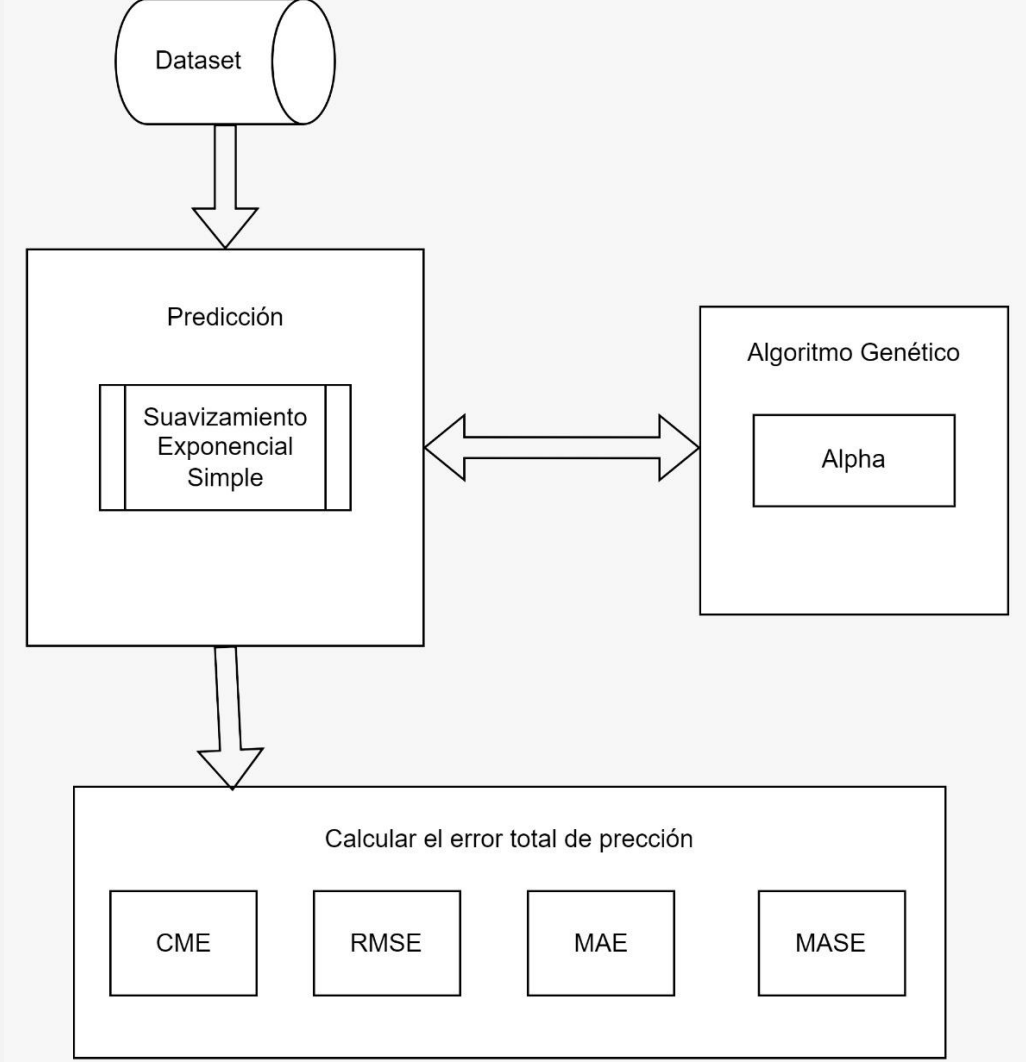
```
> var_normal(data1)#DATASET 1
[1] "p-value max =  0.00492059908350963"
[1] "Índice de columna:  17"
[1] "Nombre de columna:  Slowness in traffic (%)"
> var_normal(data2)#DATASET 2
[1] "p-value max =  3.689629612142e-12"
[1] "Índice de columna:  1"
[1] "Nombre de columna:  BUDAPEST"
> var_normal(data3)#DATASET 3
[1] "p-value max =  1.96679782451628e-45"
[1] "Índice de columna:  10"
[1] "Nombre de columna:  PT08.S5(O3)"
>
```



03

APLICACIÓN DEL ALGORITMO GENÉTICO CON SUAVIZAMIENTO EXPONENCIAL

DIAGRAMA DE BLOQUES



```r
optimizar_ga <- function(
    funcion_objetivo,
    n_variables,
    optimizacion,
    limite_inf          = NULL,
    limite_sup          = NULL,
    n_poblacion         = 20,
    n_generaciones      = 50,
    elitismo            = 0.1,
    prob_mut            = 0.01,
    distribucion        = "uniforme",
    media_distribucion = 1,
    sd_distribucion     = 1,
    min_distribucion    = -1,
    max_distribucion    = 1,
    metodo_seleccion    = "tournament",
    metodo_cruce        = "uniforme",
    parada_temprana     = FALSE,
    rondas_parada       = NULL,
    tolerancia_parada   = NULL,
    verbose             = 1,
    y,
    ...) {
```

CÓDIGO BASE

```
# ARGUMENTOS
# ============================================================================
# funcion_objetivo: nombre de la función que se desea optimizar. Debe de haber
#                   sido definida previamente.
# n_variables:      longitud de los individuos.
# optimizacion:     "maximizar" o "minimizar". Dependiendo de esto, la relación
#                   del fitness es directamente o indirectamente proporcional al
#                   valor de la función.
# limite_inf:       vector con el límite inferior de cada variable. Si solo se
#                   quiere imponer límites a algunas variables, emplear NA para
#                   las que no se quiere acotar.
# limite_sup:       vector con el límite superior de cada variable. Si solo se
#                   quiere imponer límites a algunas variables, emplear NA para
#                   las que no se quieren acotar.
# n_poblacion:      número total de individuos de la población.
# n_generaciones:   número total de generaciones creadas.
# elitismo:         porcentaje de mejores individuos de la población actual que
#                   pasan directamente a la siguiente población.
# prob_mut:         probabilidad que tiene cada posición del individuo de mutar.
# distribucion:     distribución de la que obtener el factor de mutación. Puede
#                   ser: "normal", "uniforme" o "aleatoria".
# media_distribucion: media de la distribución si se selecciona distribucion="normal".
# sd_distribucion:  desviación estándar de la distribución si se selecciona
#                   distribucion="normal".
# min_distribucion: mínimo la distribución si se selecciona distribucion="uniforme".
# max_distribucion: máximo la distribución si se selecciona distribucion="uniforme".
# metodo_seleccion: método para establecer la probabilidad de selección. Puede
#                   ser: "ruleta", "rank" o "tournament".
# metodo_cruce: método para cruzar los individuos. Puede ser: "uniforme",
#                  "punto_simple".
# parada_temprana:  si durante las últimas "rondas_parada" generaciones la diferencia
#                   absoluta entre mejores individuos no es superior al valor de
#                  "tolerancia_parada", se detiene el algoritmo y no se crean
#                   nuevas generaciones.
```

```
# rondas_parada:     número de generaciones consecutivas sin mejora mínima para que
#                    se active la parada temprana.
# tolerancia_parada: valor mínimo que debe tener la diferencia de generaciones
#                     consecutivas para considerar que hay cambio.
# verbose:           Nivel de detalle para que se imprima por pantalla el
#                    resultado de cada paso del algoritmo (0, 1, 2)

# RETORNO
# ==============================================================================
# La función devuelve una lista con 5 elementos:
# fitness:            una lista con el fitness del mejor individuo de cada
#                     generación.
# mejores_individuos: una lista con la combinación de predictores del mejor
#                     individuo de cada generación.
# mejor_individuo:    combinación de predictores del mejor individuo encontrado
#                     en todo el proceso.
# diferencia_abs:     una lista con la diferencia absoluta entre el fitness
#                     del mejor individuo de generaciones consecutivas.
# df_resultados:      un dataframe con todos los resultados anteriores.
```

04

EXPERIMENTOS

Los algoritmos utilizados fueron:

- Fuerza Bruta
- Algoritmo Genético: Método 1
- Algoritmo Genético: Método 2

Mejora una salida mediante el uso de una entrada específica.

```
#         DEFINICIÓN DE FUNCIÓN OBJETIVO
funcion <- function(x1){
  x1 =NULL
  sua = HoltWinters(y, alpha=x1, beta=FALSE, gamma=FALSE) #suavizamiento exponencial
  s1 = sua$fitted[,1] #valores del suavizamiento exponencial
  err_pr_sua = y[2:(length(y))]-s1 #error del pronostico con suavizamiento
  err_pr_sua2 = err_pr_sua^2
  CME_s = sum(err_pr_sua2)/length(s1)
}
```

Experimento 1

Resultados del primer experimento con el vector de pruebas

Mediante el algoritmo de fuerza bruta

```
"Saltos de = 0.01"
"Alpha Optimo = 0.17"
"MSE Minimo = 8.96062474006957"
"MAE = 3.709082437463"
"RMSE = 4.01531802902107"
"MASE = 1.00245471282784"
"Tiempo de Ejecucion:"
```

Experimento 1

MEDIANTE EL ALGORITMO GENÉTICO: MÉTODO 1

```
------------------------
Optimización finalizada
------------------------
Duración selección  = Time difference of 0.259022 secs
Número generaciones = 11
Límite inferior     = 0
Límite superior     = 1
Optimización        = minimizar
Valor alpha Óptimo     = 0.1680415
MSE Óptimo encontrado   = 8.959887
MAE      = 2.581814
MASE      = 0.6977874
RMSE      = 2.993567
```

Experimento 1

MEDIANTE EL ALGORITMO GENÉTICO FUNCIÓN GA - R: MÉTODO 2

```
> tic()
> GA <- ga(type="real-valued", fitness = funcion , lo
GA | iter = 1 | Mean = 8.959887 | Best = 8.959887
> print(paste("CME Óptimo",GA@fitnessValue))
[1] "CME Óptimo 8.95988720167204"
> toc()
0.03 sec elapsed
```

Experimento 2

Resultados del segundo experimento con el primer dataset

Mediante el algoritmo de fuerza bruta

```
[1] "Saltos de = 0.01"
[1] "Alpha Optimo = 0.71"
[1] "MSE Minimo = 1926.42500651869"
[1] "MAE = 25.6411652317369"
[1] "RMSE = 45.06363144043 08"
[1] "MASE = 1.0009612491403"
[1] "Tiempo de Ejecucion:"
0.03 sec elapsed
```

Experimento 2

MEDIANTE EL ALGORITMO GENÉTICO: MÉTODO 1

```
------------------------
Optimización finalizada
------------------------
Duración selección  = Time difference of 0.2679482 secs
Número generaciones = 11
Límite inferior     = 0
Límite superior     = 1
Optimización        = minimizar
Valor alpha Óptimo     = 0.7087271
MSE Óptimo encontrado   = 1926.407
MAE     = 27.28898
MASE     = 1.065287
RMSE     = 43.89123
```

Experimento 2

MEDIANTE EL ALGORITMO GENÉTICO FUNCIÓN GA - R: MÉTODO 2

```
> GA <- ga(type="real-valued", fitness = funcion ,
GA | iter = 1 | Mean = 1926.407 | Best = 1926.407
> print(paste("CME Óptimo = ",GA@fitnessValue))
[1] "CME Óptimo =  1926.40667939797"
> toc()
0.03 sec elapsed
```

Experimento 3

Resultados del tercer experimento con el segundo dataset

Mediante el algoritmo de fuerza bruta

```
[1] "Saltos de = 0.01"
[1] "Alpha Optimo = 0.41"
[1] "MSE Minimo = 2912.32419690888"
[1] "MAE = 42.4750113369137"
[1] "RMSE = 64.2795404232861"
[1] "MASE = 0.993656914486014"
[1] "Tiempo de Ejecucion:"
0.03 sec elapsed
```

Experimento 3

MEDIANTE EL ALGORITMO GENÉTICO: MÉTODO 1

```
------------------------
Optimización finalizada
------------------------
Duración selección  = Time difference of 0.266005 secs
Número generaciones = 11
Límite inferior     = 0
Límite superior     = 1
Optimización        = minimizar
Valor alpha Óptimo     = 0.3999057
MSE Óptimo encontrado   = 2912.263
MAE      = 36.61433
MASE     = 0.8565526
RMSE     = 53.96825
```

Experimento 3

MEDIANTE EL ALGORITMO GENÉTICO FUNCIÓN GA - R: MÉTODO 2

```
> GA <- ga(type="real-valued", fitness = funcion , lower
GA | iter = 1 | Mean = 2912.263 | Best = 2912.263
> print(paste("CME Óptimo = ",GA@fitnessValue))
[1] "CME Óptimo =  2912.26282441924"
> toc()
0.04 sec elapsed
```

Experimento 4

Resultados del cuarto experimento con el tercer dataset

Mediante el algoritmo de fuerza bruta

```
[1] "Saltos de = 0.01"
[1] "Alpha Optimo = 0.990000000000001"
[1] "MSE Minimo = 38311.5198111981"
[1] "MAE = 129.887466966093"
[1] "RMSE = 195.733287437774"
[1] "MASE = 1.00351836774171"
[1] "Tiempo de Ejecucion:"
0.11 sec elapsed
```

Experimento 4

MEDIANTE EL ALGORITMO GENÉTICO: MÉTODO 1

```
------------------------
Optimización finalizada
------------------------
Duración selección  = Time difference of 2.786906 secs
Número generaciones = 11
Límite inferior     = 0
Límite superior     = 1
Optimización        = minimizar
Valor alpha Óptimo     = 0.9889093
MSE Óptimo encontrado   = 38158.69
MAE      = 129.9359
MASE      = 1.003892
RMSE      = 195.7773
```

Experimento 4

MEDIANTE EL ALGORITMO GENÉTICO FUNCIÓN GA - R: MÉTODO 2

```
> GA <- ga(type="real-valued", fitness = funcion ,
+           lower = 0, upper = 1, maxiter = 1, seed = 1
GA | iter = 1 | Mean = 38158.69 | Best = 38158.69
> print(paste("CME Óptimo = ",GA@fitnessValue))
[1] "CME Óptimo =  38158.6857187416"
> toc()
0.39 sec elapsed
```

TABLA CON TODAS LAS RESPUESTAS OBTENIDAS

		alpha óptimo	MCE minimo	MAE	RMSE	MASE	Tiempo (s)
Experimento 1 y = c(17,21,19, 23,18,16, 20,18,22, 20,15,22)	**Fuerza Bruta**	0.17	8.96062	3.70908	4.01532	1.00245	0.04
	Algoritmo Genético	0.16804	8.95988	2.58181	2.99356	0.69778	0.25902
	Función Ga - R		8.95989				0.03
Experimento 2 Dataset 1	**Fuerza Bruta**	0.71	1926.42500	25.64116	45.06363	1.00096	0.03
	Algoritmo Genético	0.70872	1926.407	27.28898	43.89123	1.065287	0.26794
	Función Ga - R		1926.40667				0.03
Experimento 3 Dataset 2	**Fuerza Bruta**	0.41	2912.32419	42.47501	64.27954	0.99365	0.03
	Algoritmo Genético	0.39990	2912.263	36.61433	53.96825	0.85655	0.26600
	Función Ga - R		2912.26282				0.04
Experimento 4 Dataset 3	**Fuerza Bruta**	0.99000	38311.51981	129.88746	195.73328	1.00351	0.11
	Algoritmo Genético	0.98890	38158.69	129.9359	195.7773	1.003892	2.78690
	Función Ga - R		38158.685712				0.39

05

CONCLUSIONES

- La programación del algoritmo genético cuenta con varias funciones que hacen posible la optimización del alpha a través del valor mínimo del CME. Estas funciones se ejecutan por separado. Y cuando estas ya hayan sido ejecutadas finalmente se ejecutará el algoritmo genético. Y así se obtiene el valor del alpha optimizado en función del menor CME.

- El uso del algoritmo genético para la optimización en suavizamiento exponencial, representa un reto en el entendimiento de su funcionamiento. Además, lo más importante es definir la función objetivo, es decir en base a qué optimizará el algoritmo genético.

GRACIAS POR SU ATENCIÓN