Modelos lineales - Modelos no lineales

Librerias necesarias

```
library(fpp2)
library(dplyr)
library(ggplot2)
library(reticulate)
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Datos no lineales

Primero generamos datos no lineales de la siguiente función:

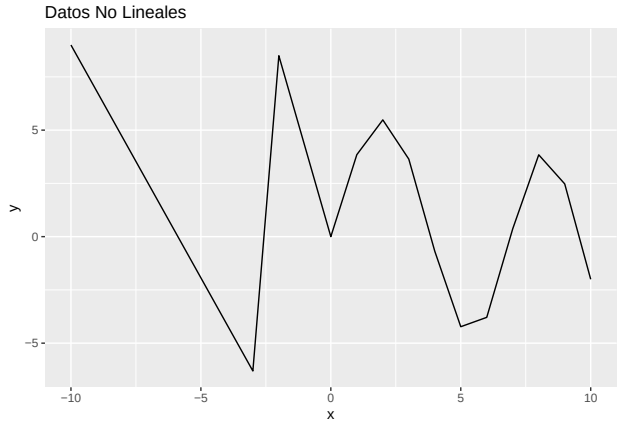
$$\begin{cases} -2.186x - 12.864 & Si \quad -\infty \leq x < -2 \\ 4.246x & Si \quad -2 \leq x < 0 \\ 10e^{-0.05x-0.5} * sen[0.03 * x + 0.7) * x] & Si \quad 0 \leq x < \infty \end{cases}$$

El correspondiente código para simular esta función es la siguiente:

```
noLinealData <- function(x) {

    if (x>=(-Inf) && x<(-2)) {
        resultado <- -2.186*x-12.864
        return(resultado)
    }

    if (x>=(-2) && x<0) {
        resultado <- -4.246*x
        return(resultado)
    }

    if (x>=0 && x<Inf) {
        resultado <- (10*exp(-0.05*x-0.5)) * sin((0.03*x+0.7)*x)
        return(resultado)
    }
}
```

Los datos para la simulación es la siguiente:

```r
x <- -10:10
data <- data.frame(x=x, y=as.numeric(lapply(x, noLinealData)))

data %>%
    ggplot(aes(x=x,y=y)) +
    geom_line() +
    ggtitle('Datos No Lineales')
```



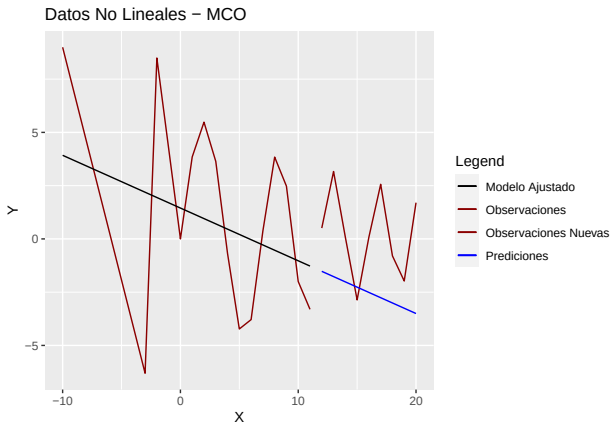
Pocos datos

```
x <- -10:20
porceTrain <- 0.7
dataTrain <- data.frame(x_train = x[1:round(length(x)*porceTrain)],
                        y_train = as.numeric(lapply(x[1:round(length(x)*porceTrain)], noLinealData)))
dataTest <- data.frame(x_test=x[(round(length(x)*porceTrain)+1):length(x)],
                       y_test=as.numeric(lapply(x[(round(length(x)*porceTrain)+1):length(x)], noLinealDa
rm(x)
```

Modelo lineal

El método de MCO intenta generalizar los datos de la función no lineal es minimizando los errores, y la función estimada es la siguiente.

```
fit <- lm(y_train~x_train, data = dataTrain)
# MODELO LINEAL
colors <- c("Observaciones" = "darkred",
            "Modelo Ajustado" = "black",
            "Observaciones Nuevas" = "darkred",
            "Prediciones" ="blue")

# GRAFICO
dataTrain %>%
    ggplot(aes(x=x_train)) +
    geom_line(aes(y=y_train, color = 'Observaciones')) +
    geom_line(aes(y=predict(fit), color='Modelo Ajustado')) +
    geom_line(aes(x=x_test,y=y_test, color='Observaciones Nuevas'), data = dataTest) +
    geom_line(aes(x=x_test,y=predict(fit, data.frame(x_train=x_test)), color='Prediciones'), data = data
    ggtitle('Datos No Lineales - MCO') +
    labs(x = "X",
         y = "Y",
         color = "Legend") +
    scale_color_manual(values = colors)
```



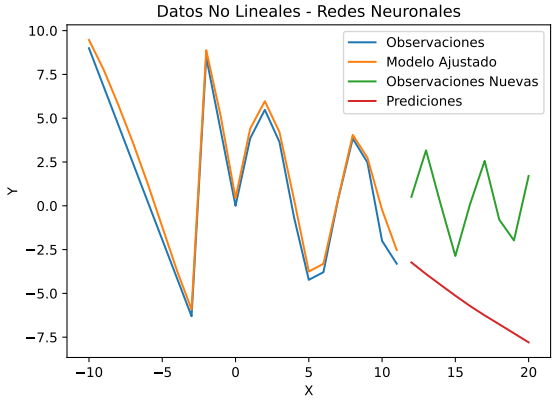
Red neuronal

```
x_train = np.array(r.dataTrain["x_train"])
y_train = np.array(r.dataTrain["y_train"])

x_test = np.array(r.dataTest["x_test"])
y_test = np.array(r.dataTest["y_test"])

model = tf.keras.Sequential([
    tf.keras.layers.Dense(50, activation='relu', input_shape=(1,)),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(1)
    ])

optimizer = tf.keras.optimizers.RMSprop(0.001)

model.compile(loss='mse',optimizer=optimizer,metrics=['mae', 'mse'])

model.fit(x_train, y_train, epochs=1000)
```

```
y_adj = model.predict(x_train)
```

```
y_adj = y_adj.reshape(-1)
y_predict = model.predict(x_test)
```

```
y_predict = y_predict.reshape(-1)


# GRAFICO
plt.clf()
plt.plot(x_train, y_train, label ='Observaciones')
plt.plot(x_train, y_adj, '-', label ='Modelo Ajustado')
plt.plot(x_test, y_test, label ='Observaciones Nuevas')
plt.plot(x_test, y_predict, label ='Prediciones')

plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title('Datos No Lineales - Redes Neuronales')
plt.show()
```


Datos No Lineales - Redes Neuronales
Observaciones
Modelo Ajustado
Observaciones Nuevas
Prediciones
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
Y
−10
−5
0
5
10
15
20
X

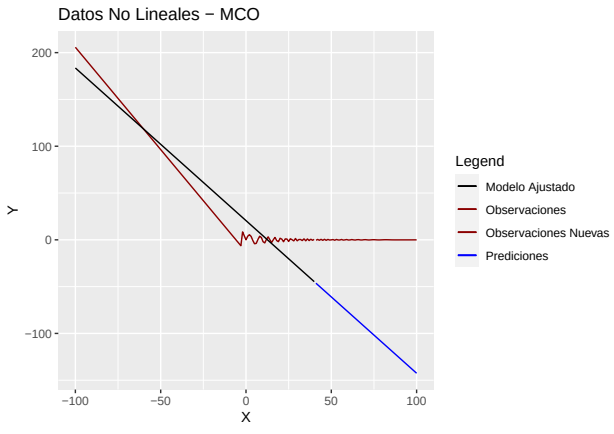
Muchos datos

```
x <- -100:100
porceTrain <- 0.7
dataTrain <- data.frame(x_train = x[1:round(length(x)*porceTrain)],
                        y_train = as.numeric(lapply(x[1:round(length(x)*porceTrain)], noLinealData)))
dataTest <- data.frame(x_test=x[(round(length(x)*porceTrain)+1):length(x)],
                       y_test=as.numeric(lapply(x[(round(length(x)*porceTrain)+1):length(x)], noLinealDa
rm(x)
```

Modelo lineal

El método de MCO intenta generalizar los datos de la función no lineal es minimizando los errores, y la función estimada es la siguiente.

```
fit <- lm(y_train~x_train, data = dataTrain)
# MODELO LINEAL
colors <- c("Observaciones" = "darkred",
            "Modelo Ajustado" = "black",
            "Observaciones Nuevas" = "darkred",
            "Prediciones" ="blue")

# GRAFICO
dataTrain %>%
    ggplot(aes(x=x_train)) +
    geom_line(aes(y=y_train, color = 'Observaciones')) +
    geom_line(aes(y=predict(fit), color='Modelo Ajustado')) +
    geom_line(aes(x=x_test,y=y_test, color='Observaciones Nuevas'), data = dataTest) +
    geom_line(aes(x=x_test,y=predict(fit, data.frame(x_train=x_test)), color='Prediciones'), data = data
    ggtitle('Datos No Lineales - MCO') +
    labs(x = "X",
         y = "Y",
         color = "Legend") +
    scale_color_manual(values = colors)
```



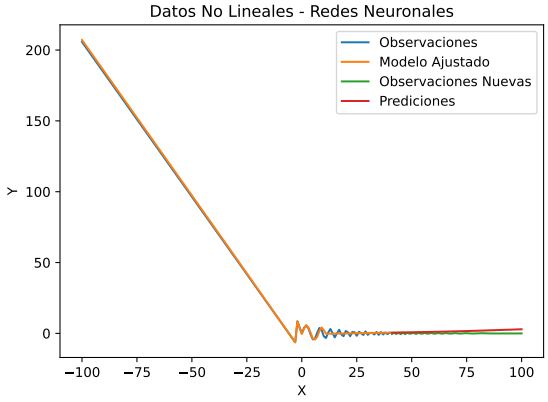
Red neuronal

```
x_train = np.array(r.dataTrain["x_train"])
y_train = np.array(r.dataTrain["y_train"])

x_test = np.array(r.dataTest["x_test"])
y_test = np.array(r.dataTest["y_test"])

model = tf.keras.Sequential([
    tf.keras.layers.Dense(50, activation='relu', input_shape=(1,)),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(1)
    ])

optimizer = tf.keras.optimizers.RMSprop(0.001)

model.compile(loss='mse',optimizer=optimizer,metrics=['mae', 'mse'])

model.fit(x_train, y_train, epochs=1000)
```

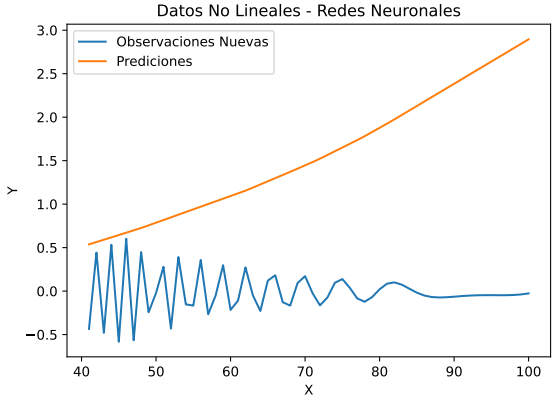
```
y_adj = model.predict(x_train)
```

```
y_adj = y_adj.reshape(-1)
y_predict = model.predict(x_test)
```

```
y_predict = y_predict.reshape(-1)


# GRAFICO
plt.clf()
plt.plot(x_train, y_train, label ='Observaciones')
plt.plot(x_train, y_adj, '-', label ='Modelo Ajustado')
plt.plot(x_test, y_test, label ='Observaciones Nuevas')
plt.plot(x_test, y_predict, label ='Prediciones')

plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title('Datos No Lineales - Redes Neuronales')
plt.show()
```



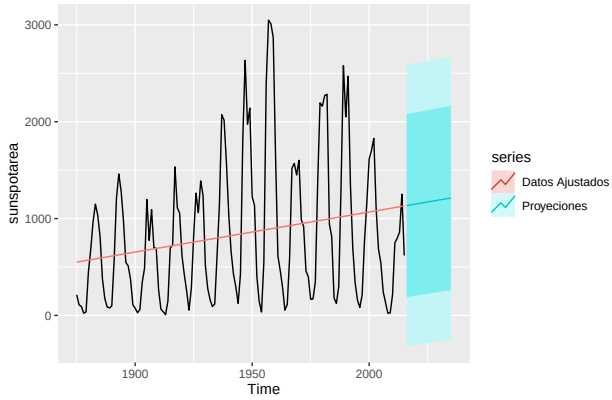

```
# GRAFICO PROYECTADO
plt.clf()
plt.plot(x_test, y_test, label ='Observaciones Nuevas')
plt.plot(x_test, y_predict, label ='Prediciones')

plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title('Datos No Lineales - Redes Neuronales')
plt.show()
```



Otro Ejemplo

Lineal

```
fitLn <- tslm(sunspotarea~trend)

autoplot(sunspotarea) +
    autolayer(fitted(fitLn), series = 'Datos Ajustados') +
    autolayer(forecast(fitLn, h = 20), series = 'Proyeciones', PI = T)
```



Red Neuronal

```
fitNnear <- nnetar(sunspotarea, lambda=0)

autoplot(sunspotarea) +
    autolayer(fitted(fitNnear), series = 'Datos Ajustados') +
    autolayer(forecast(fitNnear, h = 20), series = 'Proyeciones', PI = T)
```