Modelos NN

Jimena Murillo

2022-05-05

Paquetes

```
library(keras) # for deep learning
library(tidyverse) # general utility functions
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret) # machine learning utility functions
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tibble)
library(readr)
library(ggplot2)
library(tensorflow)
```

```
##
## Attaching package: 'tensorflow'

## The following object is masked from 'package:caret':
##
##     train
```

Construir una base con el cantón de Alajuela y partirla en train y test

```
load("C:/Users/usuario1/Desktop/CIMPA/Github_CIMPA/PRACTICA_CIMPA/base_cantones.RData")




Alajuela1 <- basecanton %>% filter(Canton == "Alajuela")

Alajuela1 <- Alajuela1%>%
  dplyr::select(Year,Month,Nino12SSTA,Nino3SSTA, Nino4SSTA, Nino34SSTA,TNA,EVI,NDVI,NDWI,LSD,LSN,Precip

  arrange(Year,Month) %>% ungroup() %>% mutate(Month=as.numeric(Month))




if(anyNA(Alajuela1)){
  Alajuela1 <- na.omit(Alajuela1)
}

#Escala


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

Alajuela1.2 <- apply(Alajuela1, 2, normalize)


#Train y test

data_train1 = as.data.frame(Alajuela1.2) %>% filter(Year < 0.85)#PARA ENTRENAR HASTA 2018
data_test1 = as.data.frame(Alajuela1.2) %>% filter(Year >= 0.85)

X_train1 = as.matrix(data_train1[,-ncol(data_train1)])
y_train1 = as.matrix(data_train1[,ncol(data_train1)])

X_test1 = as.matrix(data_test1[,-ncol(data_test1)])
y_test1 = as.matrix(data_test1[,ncol(data_test1)])
```

Base de datos con lag

```
Alajuela <- basecanton %>% filter(Canton == "Alajuela") %>%

  dplyr::select(Year,Month,Nino12SSTA, Nino3SSTA, Nino4SSTA,Nino34SSTA,Nino34SSTA1, Nino34SSTA2, Nino34S

  arrange(Year,Month) %>% ungroup() %>% mutate(Month=as.numeric(Month))


if(anyNA(Alajuela)){
  Alajuela <- na.omit(Alajuela)
}

#Escala


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

Alajuela2 <- apply(Alajuela, 2, normalize)


#Train y test

data_train = as.data.frame(Alajuela2) %>% filter(Year < 0.85)#PARA ENTRENAR HASTA 2018
data_test = as.data.frame(Alajuela2) %>% filter(Year >= 0.85)

X_train = as.matrix(data_train[,-ncol(data_train)])
y_train = as.matrix(data_train[,ncol(data_train)])

X_test = as.matrix(data_test[,-ncol(data_test)])
y_test = as.matrix(data_test[,ncol(data_test)])
```

Planteamiento de modelos:

Modelos con datos simples (sin lag)

MODELO 1

```
set.seed(123)
model <- keras_model_sequential()
```

```
## Loaded Tensorflow version 2.8.0
```

```
# our input layer
model %>%
  layer_dense(input_shape = ncol(X_train1), units = 13) %>%
  layer_dense(units = 1, activation = "relu")
```

```r
# look at our model architecture
summary(model)
```

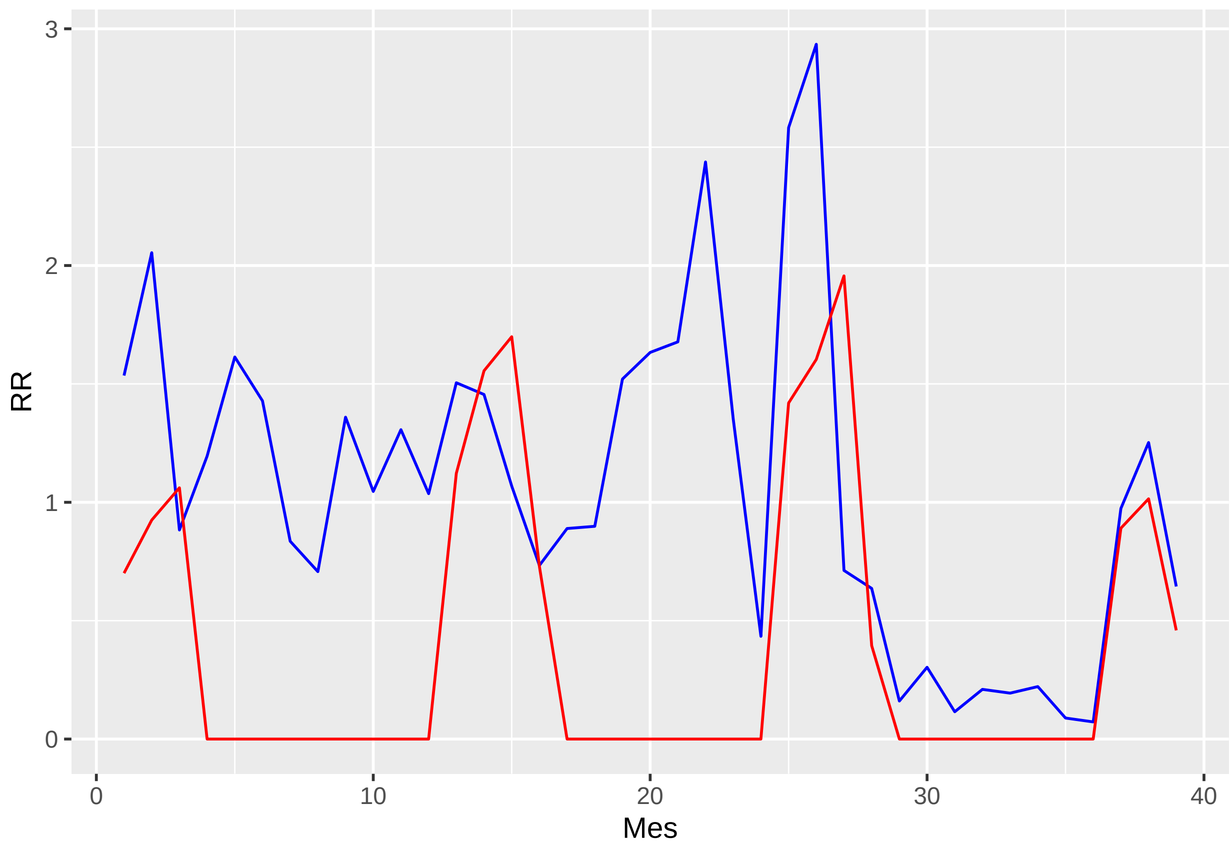
```
## Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  dense_1 (Dense)                    (None, 13)                       182
##
##  dense (Dense)                      (None, 1)                        14
##
## ================================================================================
## Total params: 196
## Trainable params: 196
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model %>% compile(loss = "mse",
                  optimizer = "adam",
                  metric = "mae")

trained_model <- model %>% fit(
  x = X_train1, # sequence we're using for prediction
  y = y_train1, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model %>% evaluate(X_test1, y_test1)
```

```
##       loss        mae
## 0.03554501 0.14927329
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

results = model %>% predict(X_test1)
results = denorm(results, max[length(Alajuela1)], min[length(Alajuela1)])

data = cbind(results, Alajuela1[197:235,length(Alajuela1)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))
```

```r
p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue")
p <- p +  geom_line(
    aes(x = Mes, y = Resultados),
    colour = "red")

print(p)
```



MODELO 2

Se agrega una capa

```r
set.seed(123)
model2 <- keras_model_sequential()
# our input layer
model2 %>%
  layer_dense(input_shape = ncol(X_train1), units = 13) %>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model2)
```

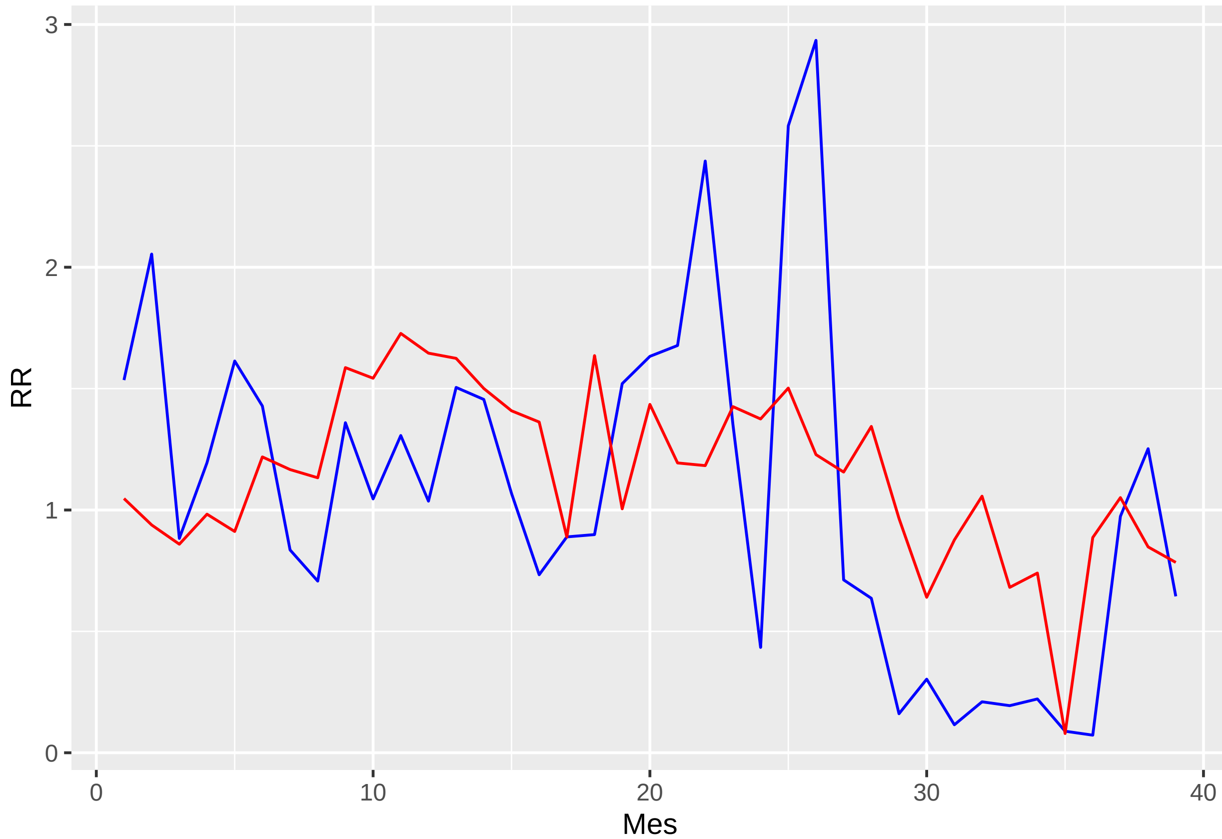
```
## Model: "sequential_1"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
```

```
## ================================================================================
##  dense_4 (Dense)                    (None, 13)                       182
##
##  dense_3 (Dense)                    (None, 8)                        112
##
##  dense_2 (Dense)                    (None, 1)                        9
##
## ================================================================================
## Total params: 303
## Trainable params: 303
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model2 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model2 <- model2 %>% fit(
  x = X_train1, # sequence we're using for prediction
  y = y_train1, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 60, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model2 %>% evaluate(X_test1, y_test1)
```

```
##                loss mean_absolute_error
##          0.01447261          0.09667838
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela1,2,max)
min <- apply(Alajuela1,2,min)

results = model2 %>% predict(X_test1)
results = denorm(results, max[length(Alajuela1)], min[length(Alajuela1)])

data = cbind(results, Alajuela1[197:nrow(Alajuela1),length(Alajuela1)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



En este modelo se observa una reducción del error cuadrado medio.

Modelo con lag:

MODELO 3

NN creada con las nuevas variables lag, se ajusta el dropout, y unidades a lo que generó mejores resultados.

```
set.seed(123)
model3 <- keras_model_sequential()
# our input layer
model3 %>%
  layer_dense(input_shape = ncol(X_train), units = 32) %>%
  layer_dropout(rate = 0.2)%>%
  layer_dense(units = 16, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model3)
```

```
## Model: "sequential_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
```

```
##  dense_7 (Dense)                     (None, 32)                      1056
##
##  dropout (Dropout)                   (None, 32)                      0
##
##  dense_6 (Dense)                     (None, 16)                      528
##
##  dense_5 (Dense)                     (None, 1)                       17
##
## ==================================================================================
## Total params: 1,601
## Trainable params: 1,601
## Non-trainable params: 0
## __________________________________________________________________________________
```

```
model3 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model3 <- model3 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model3 %>% evaluate(X_test, y_test)
```

```
##                loss mean_absolute_error
##          0.01758178          0.10515457
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model3 %>% predict(X_test)
results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

#Escala

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line(aes(x = Mes, y = Resultados),colour = "red")

print(p)
```

Se construye un modelo con rnn:

MODELO 4

```
set.seed(123)
model4 <- keras_model_sequential()
# our input layer
model4 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train),1), activation='relu') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model4)
```

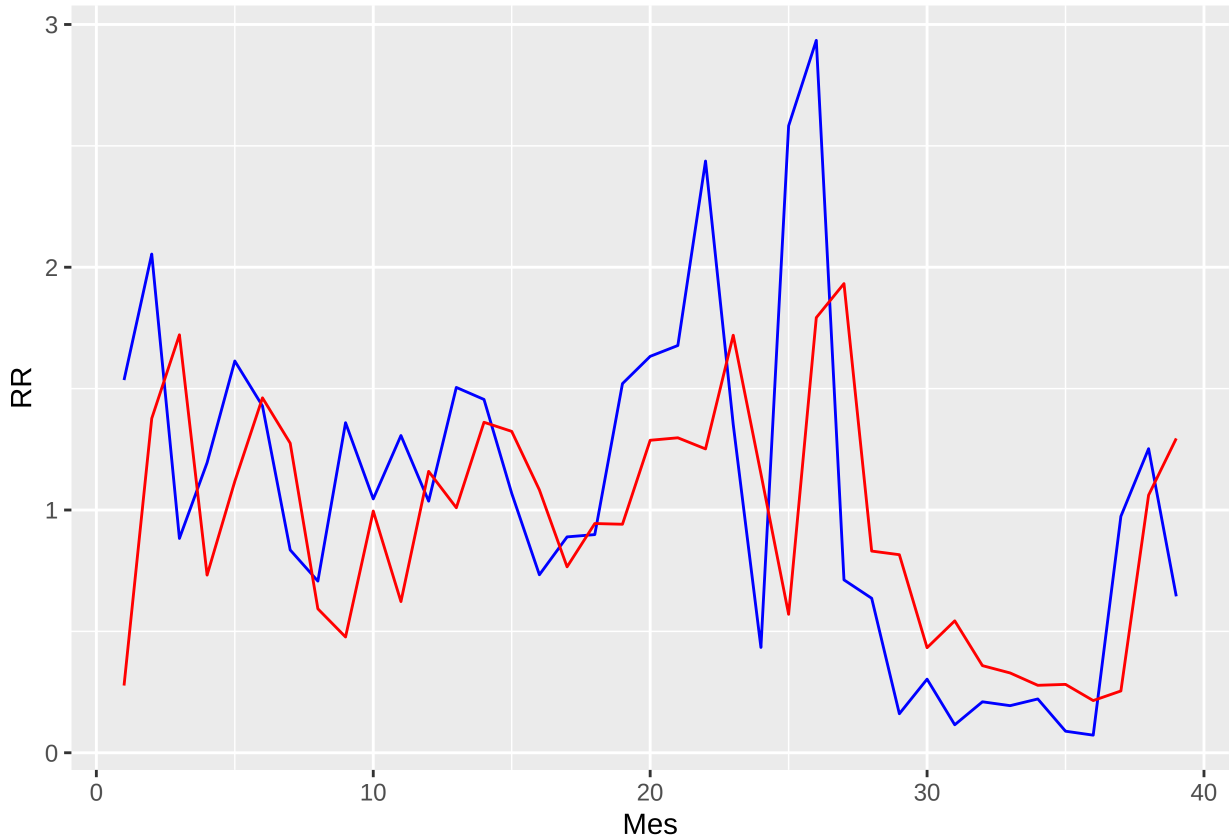
```
## Model: "sequential_3"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  simple_rnn (SimpleRNN)             (None, 24)                        624
##
```

```
##  dropout_1 (Dropout)                (None, 24)                     0
##
##  dense_9 (Dense)                    (None, 12)                     300
##
##  dense_8 (Dense)                    (None, 1)                      13
##
## ================================================================================
## Total params: 937
## Trainable params: 937
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model4 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model4 <- model4 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2,
  shuffle = F) # how much data to hold out for testing as we go along

model4 %>% evaluate(X_test, y_test)
```

```
##               loss mean_absolute_error
##         0.01531897          0.09312909
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model4 %>% predict(X_test)

results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



El modelo anterior ajusta muy bien; sin embargo, está basándose casi completamente en RR lag, es autoregresivo.

Modelos sin variable RRl1

MODELO 5

```
set.seed(123)
model5 <- keras_model_sequential()
# our input layer
model5 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train)-1,1), activation='tanh') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 1, activation = "sigmoid")


# look at our model architecture
summary(model5)
```

```
## Model: "sequential_4"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
```

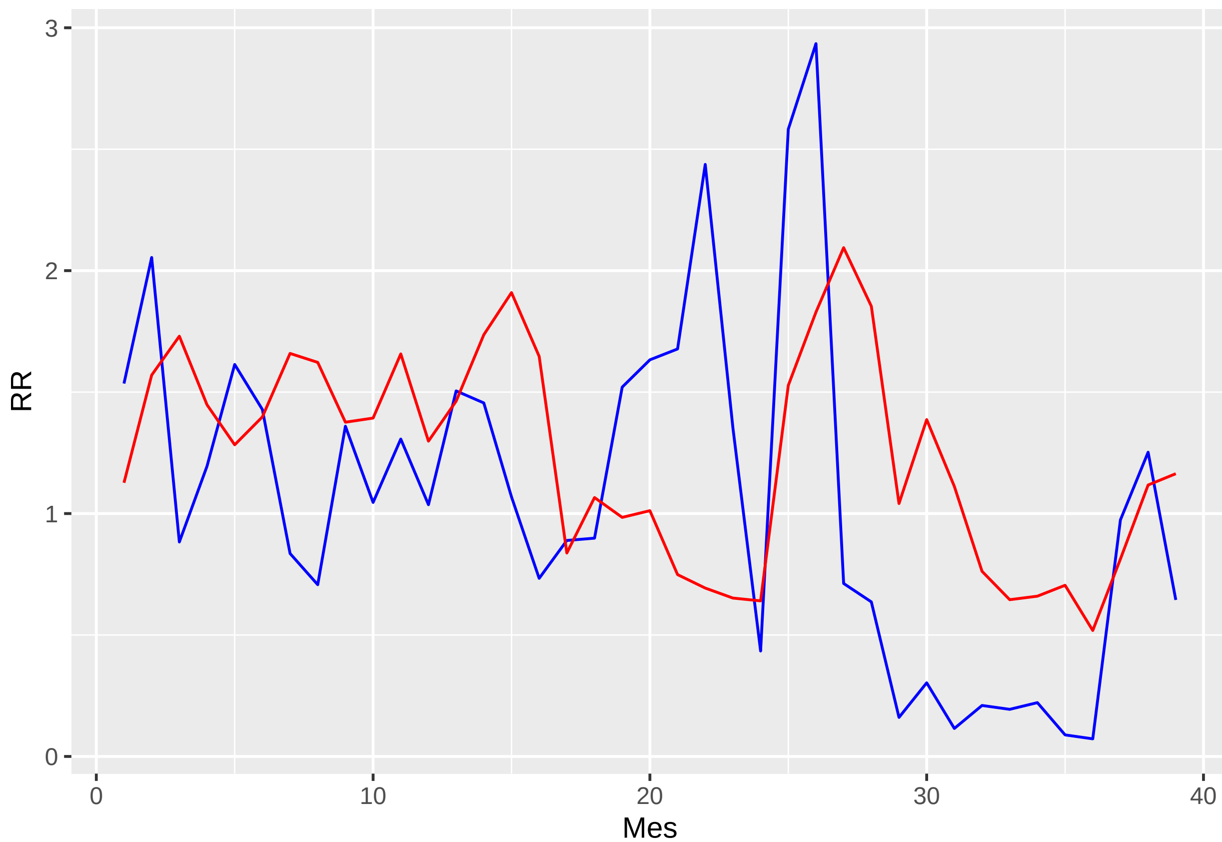
```
## ================================================================================
##  simple_rnn_1 (SimpleRNN)           (None, 24)                    624
##
##  dropout_3 (Dropout)                (None, 24)                    0
##
##  dense_12 (Dense)                   (None, 12)                    300
##
##  dense_11 (Dense)                   (None, 8)                     104
##
##  dropout_2 (Dropout)                (None, 8)                     0
##
##  dense_10 (Dense)                   (None, 1)                     9
##
## ================================================================================
## Total params: 1,037
## Trainable params: 1,037
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model5 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model5 <- model5 %>% fit(
  x = X_train[,-32], # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.1,
  shuffle = F) # how much data to hold out for testing as we go along

model5 %>% evaluate(X_test[,-32], y_test)
```

```
##                loss mean_absolute_error
##          0.01884599          0.11333553
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model5 %>% predict(X_test[,-32])

results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)
```

```
Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



El ajuste no parece ser tan bueno como el modelo autoregresivo, pero no es un mal ajuste y no tiene autoregresión.

También se crea un modelo NN sin la variable de RRl1:

MODELO 6

```
set.seed(123)
model6 <- keras_model_sequential()
# our input layer
model6 %>%
  layer_dense(input_shape = ncol(X_train)-1, units = 32) %>%
  layer_dense(units = 16, activation = "tanh")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dense(units = 4, activation = "relu")%>%
  layer_dropout(rate = 0.15)%>%
  layer_dense(units = 1, activation = "sigmoid")
```

```r
# look at our model architecture
summary(model6)
```

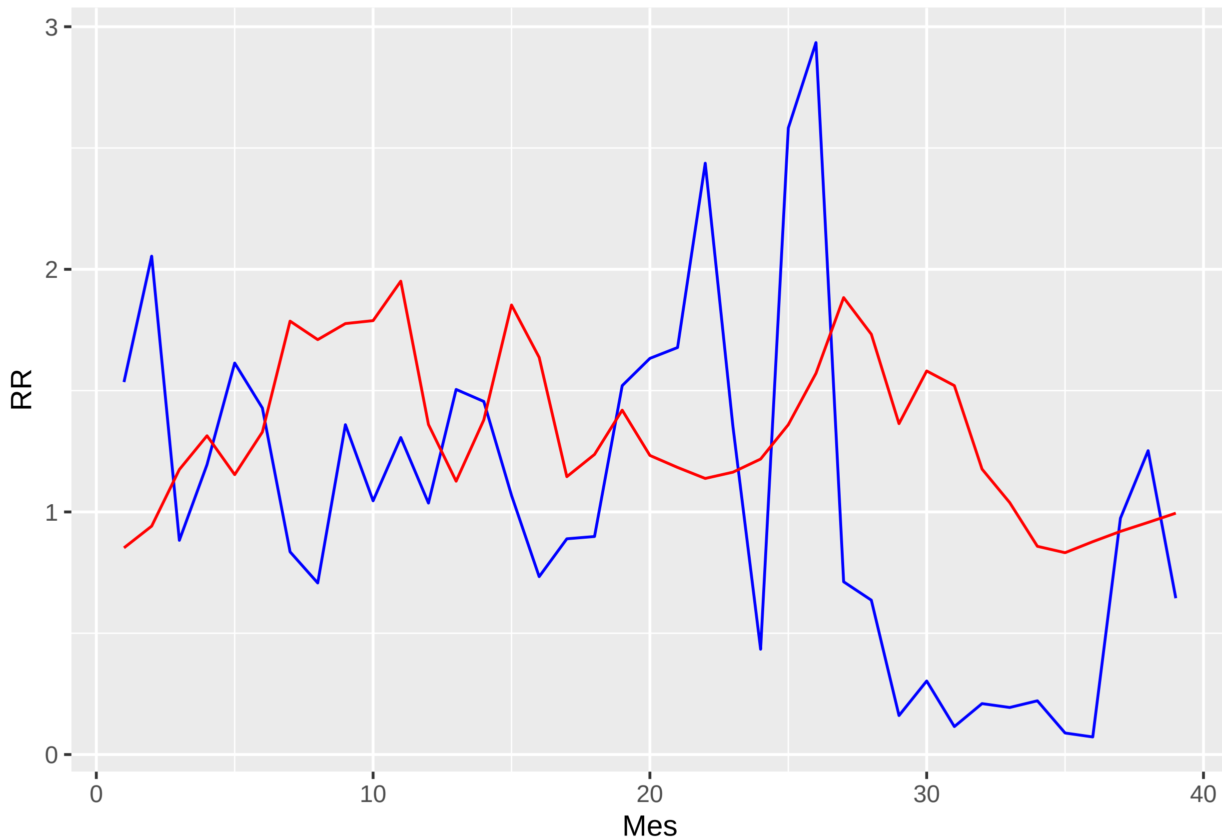
```
## Model: "sequential_5"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_17 (Dense)                   (None, 32)                      1024
##
##  dense_16 (Dense)                   (None, 16)                      528
##
##  dense_15 (Dense)                   (None, 8)                       136
##
##  dense_14 (Dense)                   (None, 4)                       36
##
##  dropout_4 (Dropout)                (None, 4)                       0
##
##  dense_13 (Dense)                   (None, 1)                       5
##
## ================================================================================
## Total params: 1,729
## Trainable params: 1,729
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model6 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model6 <- model6 %>% fit(
  x = X_train[,-32], # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 80, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model6 %>% evaluate(X_test[,-32], y_test1)
```

```
##                loss mean_absolute_error
##          0.02258279          0.12873666
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model6 %>% predict(X_test[,-32])
results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])
```

```
data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



Ahora se prueba utilizar la estructura de los modelos anteriores, pero con la variable RRl1

Nuevos modelos con RRl1

MODELO 7

Tiene la estructura del modelo 5

```
set.seed(123)
model7 <- keras_model_sequential()
# our input layer
model7 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train),1), activation='tanh') %>%
  layer_dropout(rate = 0.4)%>%
```

```r
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 1, activation = "sigmoid")


# look at our model architecture
summary(model7)
```

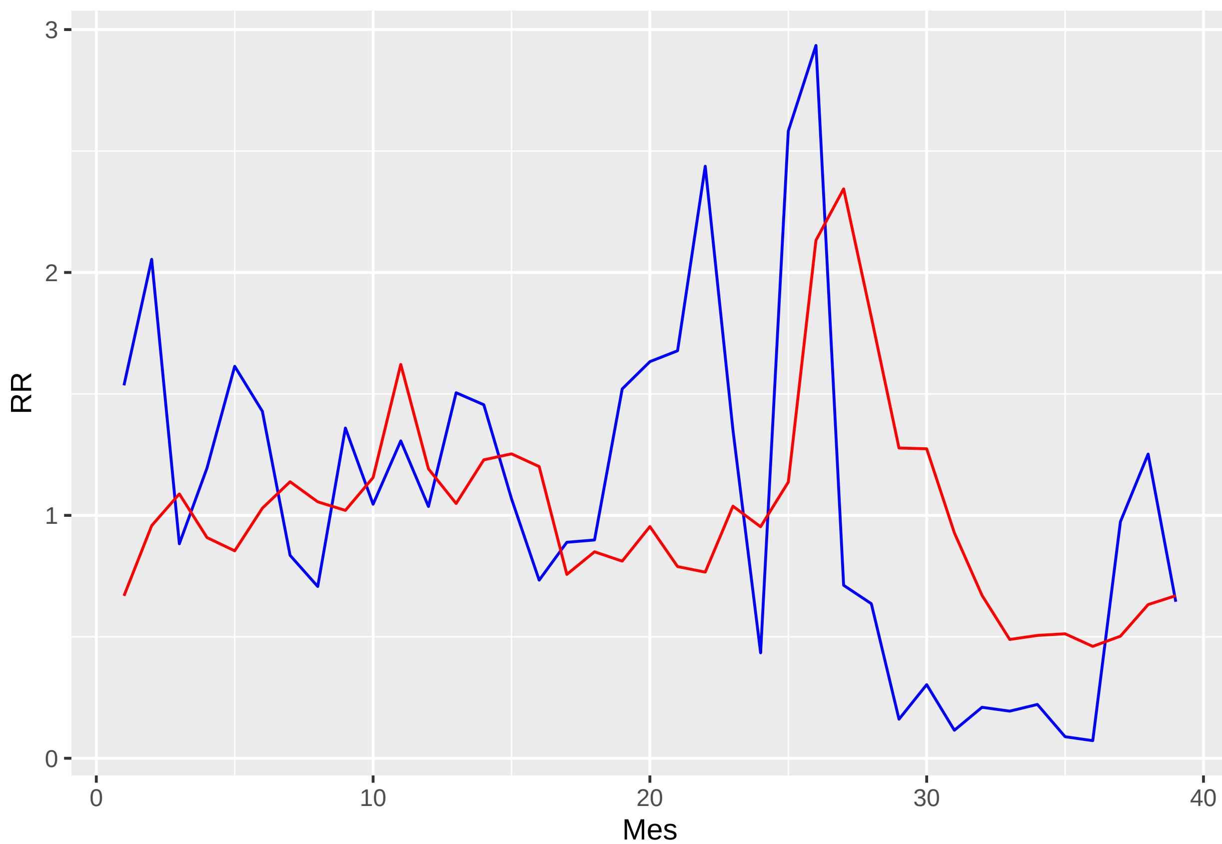
```
## Model: "sequential_6"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  simple_rnn_2 (SimpleRNN)           (None, 24)                      624
##
##  dropout_6 (Dropout)                (None, 24)                      0
##
##  dense_20 (Dense)                   (None, 12)                      300
##
##  dense_19 (Dense)                   (None, 8)                       104
##
##  dropout_5 (Dropout)                (None, 8)                       0
##
##  dense_18 (Dense)                   (None, 1)                       9
##
## ================================================================================
## Total params: 1,037
## Trainable params: 1,037
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model7 %>% compile(loss = "mean_squared_error",
                   optimizer = "adam",
                   metric = "mean_absolute_error")

trained_model7 <- model7 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 80, # how many times we'll look @ the whole dataset
  validation_split = 0.1,
  shuffle = F) # how much data to hold out for testing as we go along

model7 %>% evaluate(X_test, y_test)
```

```
##                loss mean_absolute_error
##          0.01840702          0.10974184
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
```

```
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model7 %>% predict(X_test)

results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



MODELO 8

Tiene la estructura del modelo 6

```
set.seed(123)
model8 <- keras_model_sequential()
```

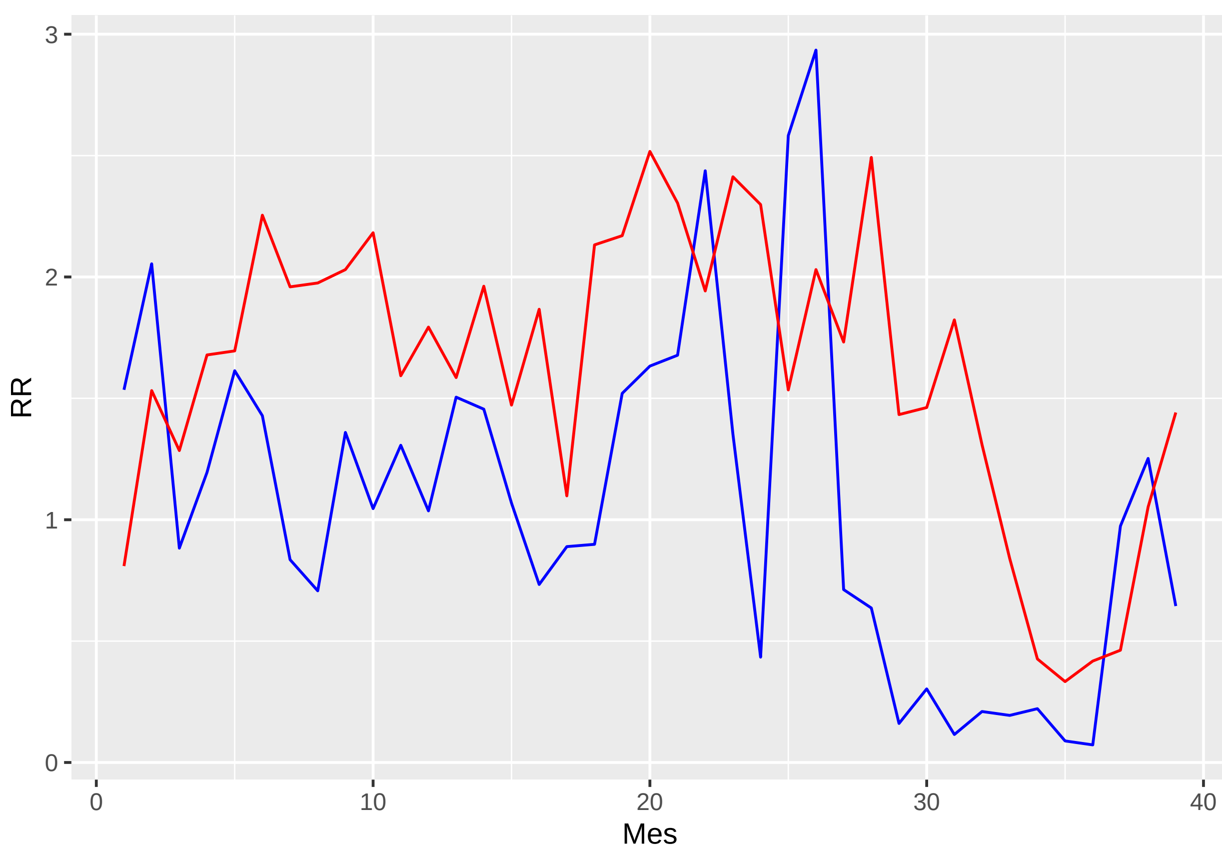
```r
# our input layer
model8 %>%
  layer_dense(input_shape = ncol(X_train), units = 32) %>%
  layer_dense(units = 16, activation = "tanh")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dense(units = 4, activation = "relu")%>%
  layer_dropout(rate = 0.15)%>%
  layer_dense(units = 1, activation = "sigmoid")


# look at our model architecture
summary(model8)
```

```
## Model: "sequential_7"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  dense_25 (Dense)                   (None, 32)                        1056
##
##  dense_24 (Dense)                   (None, 16)                        528
##
##  dense_23 (Dense)                   (None, 8)                         136
##
##  dense_22 (Dense)                   (None, 4)                         36
##
##  dropout_7 (Dropout)                (None, 4)                         0
##
##  dense_21 (Dense)                   (None, 1)                         5
##
## ================================================================================
## Total params: 1,761
## Trainable params: 1,761
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model8 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model8 <- model8 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 60, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model8 %>% evaluate(X_test, y_test1)
```

```
##                loss mean_absolute_error
##          0.02987122          0.14911212
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)

results = model8 %>% predict(X_test)
results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



Modelos con lag 2 y 3

Se crea la variable RRl2 y RRl3 para probar el efecto de la autoregresión con diferentes niveles de lag

```
Alajuela3 <- Alajuela %>% mutate(RRl2 = lag(RR,2), RRl3 = lag(RR,3))


if(anyNA(Alajuela3)){
  Alajuela3 <- na.omit(Alajuela3)
}


Alajuela3 <- Alajuela3 %>% dplyr::select(Year,Month,Nino12SSTA, Nino3SSTA, Nino4SSTA,Nino34SSTA,Nino34SS

  arrange(Year,Month) %>% ungroup() %>% mutate(Month=as.numeric(Month))







#Escala


normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

max <- apply(Alajuela3,2,max)
min <- apply(Alajuela3,2,min)

Alajuela3.2 <- apply(Alajuela3, 2, normalize)


#Train y test

data_train3 = as.data.frame(Alajuela3.2) %>% filter(Year < 0.85)#PARA ENTRENAR HASTA 2018
data_test3 = as.data.frame(Alajuela3.2) %>% filter(Year >= 0.85)

X_train3 = as.matrix(data_train3[,-ncol(data_train3)])
y_train3 = as.matrix(data_train3[,ncol(data_train3)])

X_test3 = as.matrix(data_test3[,-ncol(data_test3)])
y_test3 = as.matrix(data_test3[,ncol(data_test3)])
```

MODELO 9

RNN con lag 2

```
set.seed(123)
model9 <- keras_model_sequential()
# our input layer
model9 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train3)-2,1), activation='relu') %>%
```

```
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model9)
```

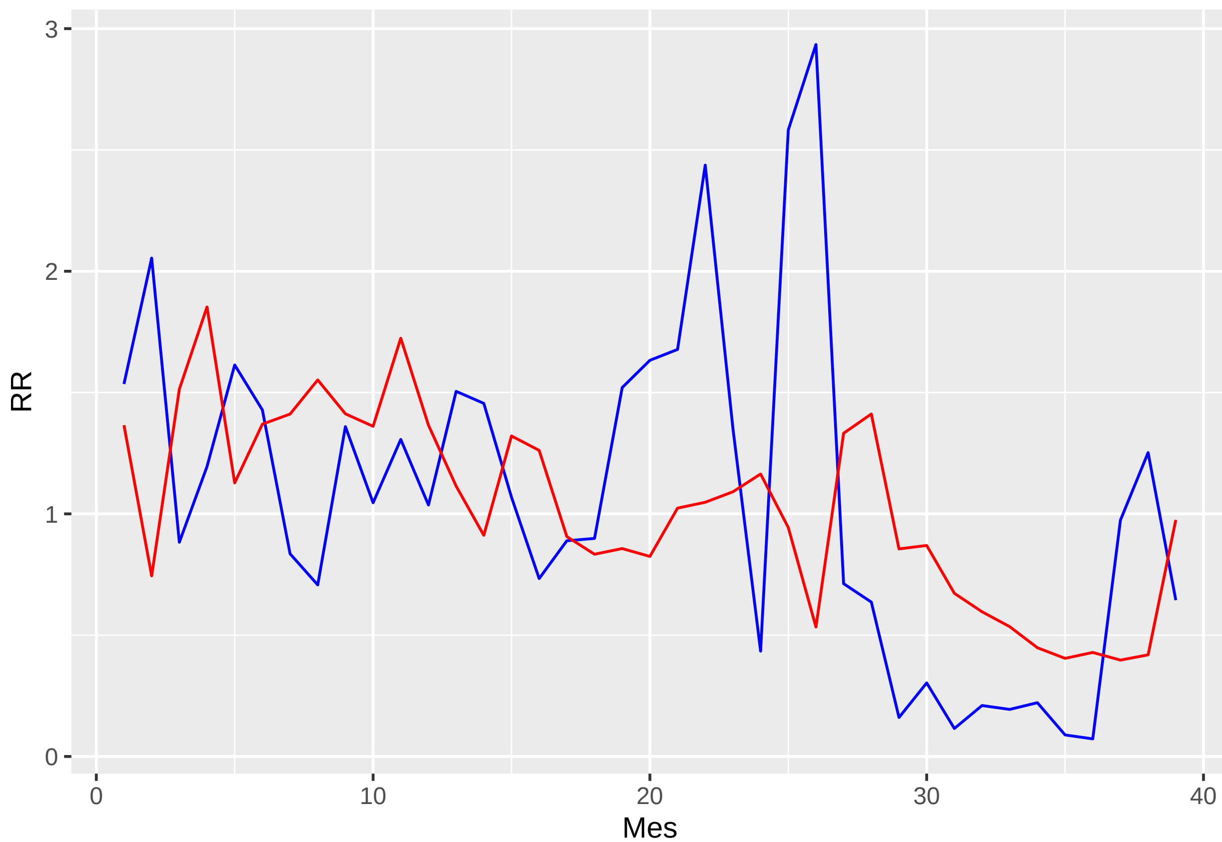
```
## Model: "sequential_8"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  simple_rnn_3 (SimpleRNN)           (None, 24)                      624
##
##  dropout_8 (Dropout)                (None, 24)                      0
##
##  dense_27 (Dense)                   (None, 12)                      300
##
##  dense_26 (Dense)                   (None, 1)                       13
##
## ================================================================================
## Total params: 937
## Trainable params: 937
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model9 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model9 <- model9 %>% fit(
  x = X_train3[,-c(32,34)], # sequence we're using for prediction
  y = y_train3, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2,
  shuffle = F) # how much data to hold out for testing as we go along

model9 %>% evaluate(X_test3[,-c(32,34)], y_test3)
```

```
##                loss mean_absolute_error
##          0.01994772          0.11152397
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela3,2,max)
min <- apply(Alajuela3,2,min)
```

```r
results = model9 %>% predict(X_test3[,-c(32,34)])

results = denorm(results, max[length(Alajuela3)], min[length(Alajuela3)])

data = cbind(results, Alajuela3[194:nrow(Alajuela3),length(Alajuela3)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



MODELO 10

RNN con lag 3

```r
set.seed(123)
model10 <- keras_model_sequential()
# our input layer
model10 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train3)-2,1), activation='relu') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
```

```r
  layer_dense(units = 1, activation = "relu")


# look at our model architecture
summary(model10)
```

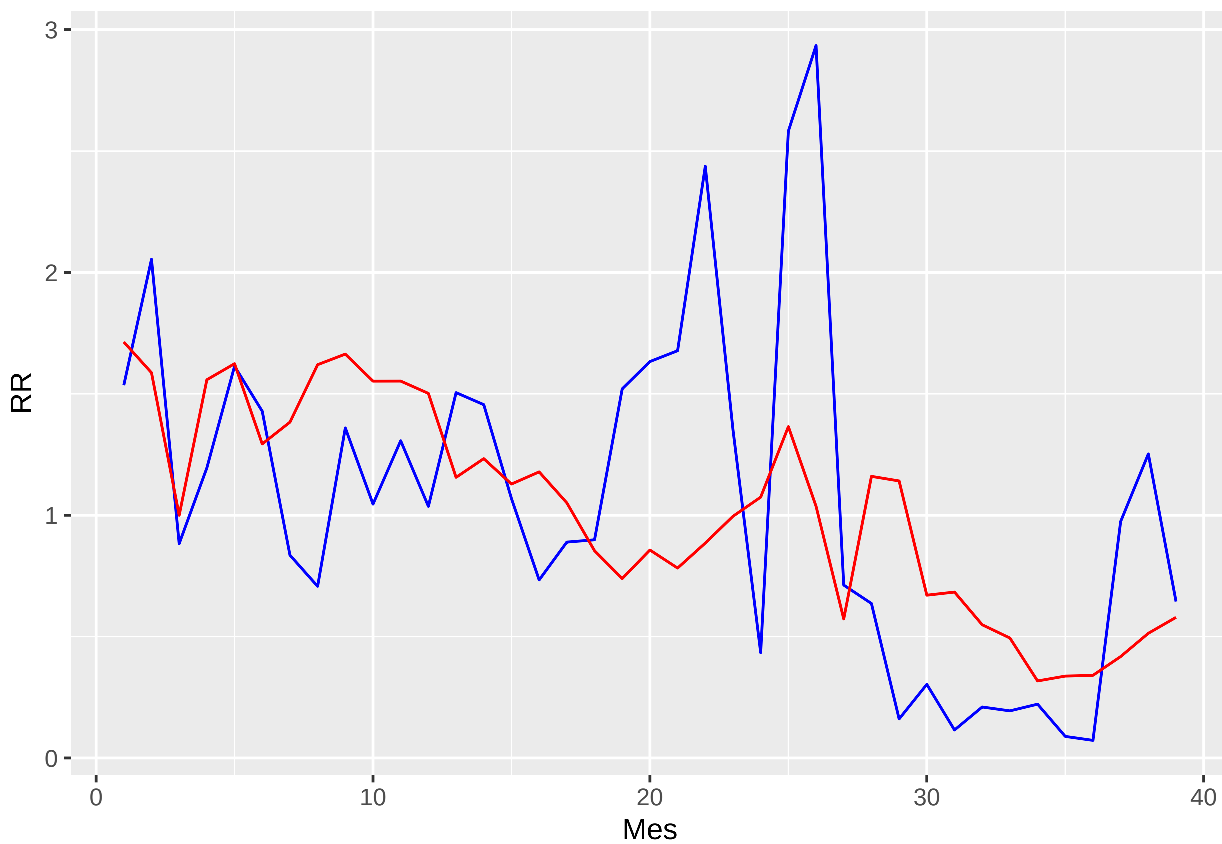
```
## Model: "sequential_9"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                      Param #
## ================================================================================
##  simple_rnn_4 (SimpleRNN)           (None, 24)                        624
##
##  dropout_9 (Dropout)                (None, 24)                        0
##
##  dense_29 (Dense)                   (None, 12)                        300
##
##  dense_28 (Dense)                   (None, 1)                         13
##
## ================================================================================
## Total params: 937
## Trainable params: 937
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model10 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model10 <- model10 %>% fit(
  x = X_train3[,-c(32,33)], # sequence we're using for prediction
  y = y_train3, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2,
  shuffle = F) # how much data to hold out for testing as we go along

model10 %>% evaluate(X_test3[,-c(32,33)], y_test3)
```

```
##                loss mean_absolute_error
##          0.01453476          0.09228259
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela3,2,max)
min <- apply(Alajuela3,2,min)

results = model10 %>% predict(X_test3[,-c(32,33)])
```

```r
results = denorm(results, max[length(Alajuela3)], min[length(Alajuela3)])

data = cbind(results, Alajuela3[194:nrow(Alajuela3),length(Alajuela3)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



MODELO 11

Modelo con los 3 RRl

```r
set.seed(123)
model11 <- keras_model_sequential()
# our input layer
model11 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train3),1), activation='relu') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 1, activation = "relu")
```

```r
# look at our model architecture
summary(model11)
```

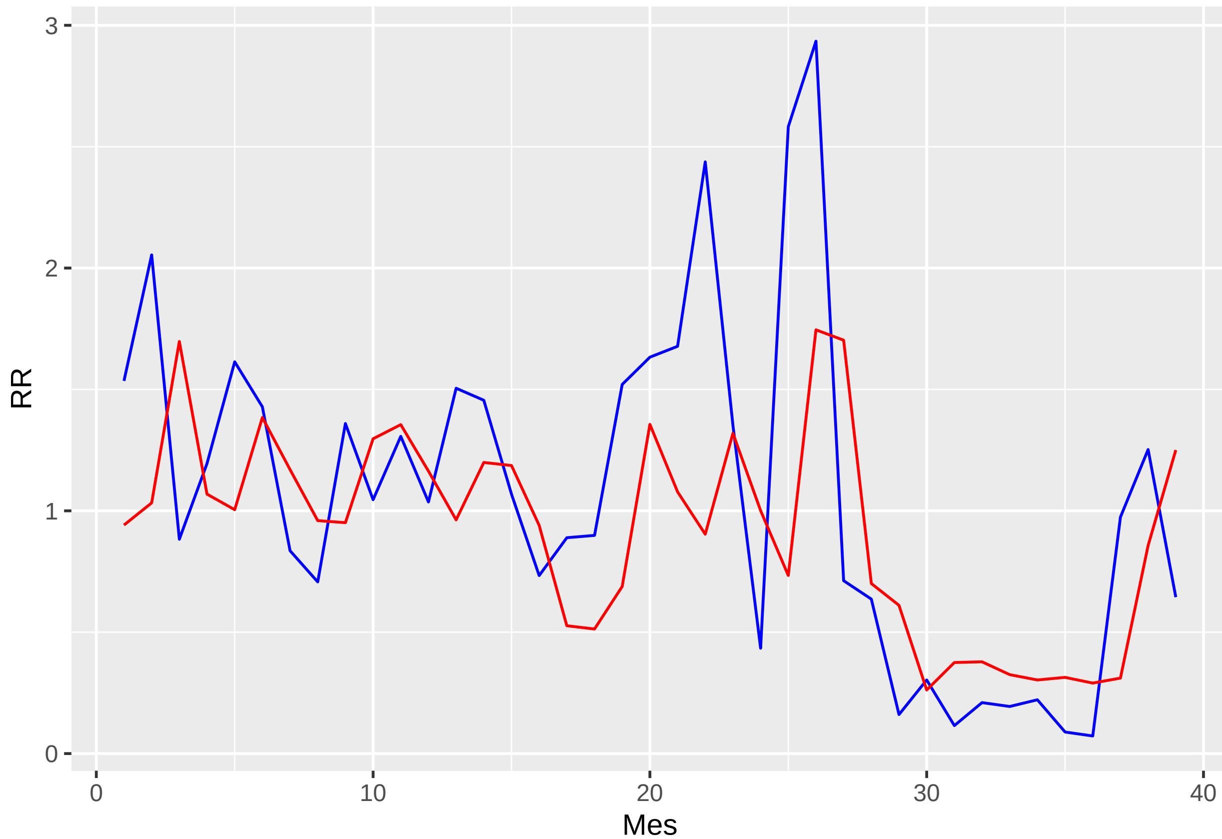
```
## Model: "sequential_10"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  simple_rnn_5 (SimpleRNN)           (None, 24)                      624
##
##  dropout_10 (Dropout)               (None, 24)                      0
##
##  dense_31 (Dense)                   (None, 12)                      300
##
##  dense_30 (Dense)                   (None, 1)                       13
##
## ================================================================================
## Total params: 937
## Trainable params: 937
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model11 %>% compile(loss = "mean_squared_error",
                   optimizer = "adam",
                   metric = "mean_absolute_error")

trained_model11 <- model11 %>% fit(
  x = X_train3, # sequence we're using for prediction
  y = y_train3, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.2,
  shuffle = F) # how much data to hold out for testing as we go along

model11 %>% evaluate(X_test3, y_test3)
```

```
##                loss mean_absolute_error
##          0.01363346          0.08656619
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela3,2,max)
min <- apply(Alajuela3,2,min)

results = model11 %>% predict(X_test3)

results = denorm(results, max[length(Alajuela3)], min[length(Alajuela3)])

data = cbind(results, Alajuela3[194:nrow(Alajuela3),length(Alajuela3)])
```

```r
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



MODELO 12

Estructura del modelo 5 con 3 rezagos

```r
set.seed(123)
model12 <- keras_model_sequential()
# our input layer
model12 %>%
  layer_simple_rnn(units = 24, input_shape = c(ncol(X_train3),1), activation='tanh') %>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 12, activation = "relu")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dropout(rate = 0.4)%>%
  layer_dense(units = 1, activation = "sigmoid")
```

```
# look at our model architecture
summary(model12)
```

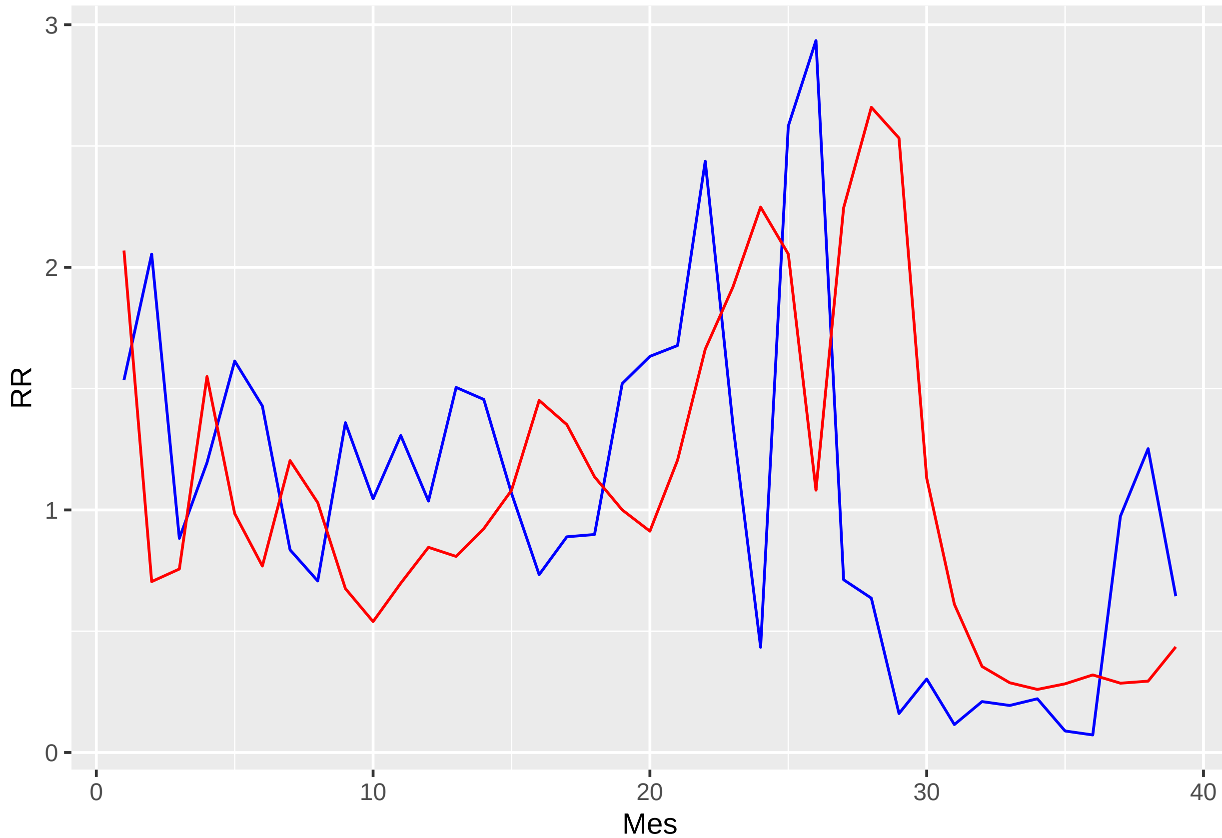
```
## Model: "sequential_11"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  simple_rnn_6 (SimpleRNN)           (None, 24)                       624
##
##  dropout_12 (Dropout)               (None, 24)                       0
##
##  dense_34 (Dense)                   (None, 12)                       300
##
##  dense_33 (Dense)                   (None, 8)                        104
##
##  dropout_11 (Dropout)               (None, 8)                        0
##
##  dense_32 (Dense)                   (None, 1)                        9
##
## ================================================================================
## Total params: 1,037
## Trainable params: 1,037
## Non-trainable params: 0
## ________________________________________________________________________________
```

```
model12 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model12 <- model12 %>% fit(
  x = X_train3, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 50, # how many times we'll look @ the whole dataset
  validation_split = 0.1,
  shuffle = F) # how much data to hold out for testing as we go along

model12 %>% evaluate(X_test3, y_test)
```

```
##                loss mean_absolute_error
##          0.02760298          0.12766711
```

```
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela3,2,max)
min <- apply(Alajuela3,2,min)

results = model12 %>% predict(X_test3)
```

```r
results = denorm(results, max[length(Alajuela3)], min[length(Alajuela3)])

data = cbind(results, Alajuela3[194:nrow(Alajuela3),length(Alajuela3)])
colnames(data) = c("Resultados", "RR")
data = as.data.frame(data)

Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```



MODELO 13

Tiene la estructura del modelo 6

```r
set.seed(123)
model13 <- keras_model_sequential()
# our input layer
model13 %>%
  layer_dense(input_shape = ncol(X_train), units = 32) %>%
  layer_dense(units = 16, activation = "tanh")%>%
  layer_dense(units = 8, activation = "relu")%>%
  layer_dense(units = 4, activation = "relu")%>%
  layer_dropout(rate = 0.15)%>%
```

```r
  layer_dense(units = 1, activation = "sigmoid")


# look at our model architecture
summary(model13)
```

```
## Model: "sequential_12"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  dense_39 (Dense)                   (None, 32)                       1056
##
##  dense_38 (Dense)                   (None, 16)                       528
##
##  dense_37 (Dense)                   (None, 8)                        136
##
##  dense_36 (Dense)                   (None, 4)                        36
##
##  dropout_13 (Dropout)               (None, 4)                        0
##
##  dense_35 (Dense)                   (None, 1)                        5
##
## ================================================================================
## Total params: 1,761
## Trainable params: 1,761
## Non-trainable params: 0
## ________________________________________________________________________________
```

```r
model13 %>% compile(loss = "mean_squared_error",
                  optimizer = "adam",
                  metric = "mean_absolute_error")

trained_model13 <- model13 %>% fit(
  x = X_train, # sequence we're using for prediction
  y = y_train, # sequence we're predicting
  batch_size = 18, # how many samples to pass to our model at a time
  epochs = 60, # how many times we'll look @ the whole dataset
  validation_split = 0.2) # how much data to hold out for testing as we go along

model13 %>% evaluate(X_test, y_test1)
```

```
##                loss mean_absolute_error
##          0.01774723          0.11293495
```

```r
#Escala

denorm <- function(x, max, min) {
  return (x*(max - min)+min)
}

max <- apply(Alajuela,2,max)
min <- apply(Alajuela,2,min)
```

```
results = model13 %>% predict(X_test)
results = denorm(results, max[length(Alajuela)], min[length(Alajuela)])

data = cbind(results, Alajuela[197:nrow(Alajuela),length(Alajuela)])
names(data) = c("Resultados", "RR")


Mes = seq(1, length(results))

p <- ggplot(data, aes(x = Mes, y = RR)) + geom_line(colour = "blue") +
  geom_line( aes(x = Mes, y = Resultados), colour = "red")

print(p)
```