Bookdown - Grupo B

Andrés Peralta Alean, David Barrera Barrera, Luis Gonzalo Guerra J

2023-06-19

Contents

Chapter 1

Propuesta

Análisis de riesgo crediticio y su importancia La evaluación del riesgo crediticio es una tarea crítica para cualquier empresa que ofrezca préstamos o créditos. El incumplimiento de los términos del préstamo o la falta de pago pueden generar grandes pérdidas financieras y afectar la estabilidad de la empresa. Es por ello que es importante contar con herramientas y técnicas que permitan evaluar el riesgo crediticio de manera eficiente.

El análisis de riesgo crediticio utiliza históricos para evaluar el comportamiento de los clientes a lo largo de varios periodos. De esta forma, se pueden obtener patrones y características que permiten segregar grupos de clientes y determinar un posible perfil de riesgo. Los resultados obtenidos a partir de este análisis pueden ayudar a la empresa a tomar decisiones más informadas y a reducir el riesgo crediticio.

Además, el análisis de riesgo crediticio permite identificar posibles oportunidades para la empresa. Por ejemplo, puede ayudar a la empresa a identificar clientes que presenten un bajo riesgo crediticio y, por lo tanto, puedan recibir préstamos con tasas de interés más bajas. De esta manera, la empresa puede aumentar su base de clientes y mejorar su rentabilidad.

En cuanto a las fuentes de información, se pueden utilizar diversas fuentes para recopilar los datos necesarios para el análisis de riesgo crediticio. Entre ellas se encuentran bases de datos públicas y privadas, encuestas, registros gubernamentales, entre otras. Es importante verificar la calidad de la información obtenida para asegurar la precisión y confiabilidad de los resultados.

En el caso específico de la empresa ABC, se cuenta con los permisos necesarios por parte del área de gestión de cobranzas y se eliminaron los datos personales de los clientes con los cuales se extrajeron las características.

En resumen, el análisis de riesgo crediticio es una técnica muy útil para evaluar el riesgo crediticio de los clientes y para identificar posibles oportunidades para

la empresa. Se pueden utilizar diversas fuentes de información para recopilar los datos necesarios para este análisis, pero es importante asegurarse de contar con los permisos necesarios y verificar la calidad de la información obtenida.

Chapter 2

Analisis Exploratorio

2.1 Introducción

En este análisis, se explorará el comportamiento de la utilización de los productos de un banco a lo largo de los años. Se examinará si ha habido un incremento en la utilización de los productos año a año, y en caso afirmativo, se intentará identificar cuáles son las razones que han llevado a este aumento.

En particular, se examinará la utilización de los productos de un banco desde el año 2018 hasta el año en curso. Se considerarán productos tales como tarjetas de crédito y crédito de consumo. Se investigará si existe un patrón estacional en la utilización de estos productos, es decir, si hay un comportamiento cíclico que se repite en determinados meses del año. Además, se estudiará la posible existencia de tendencias a largo plazo que puedan indicar un cambio en el comportamiento de los clientes del banco y que posibles factores indicen en estos cambios.

La metodología utilizada en este análisis incluye la descomposición de series de tiempo, la identificación de patrones estacionales y la realización de pruebas estadísticas para verificar la presencia de tendencias y cambios en el comportamiento de los clientes.

2.2 Explicación de la base

Partimos de una base agrupada con las siguientes variables:

Periodo - > año y mes Sub_Tipo -> tipo de producto N_Clientes -> cantidad de cleintes DIAS_DE_MORA -> días de mora de los clientes a cierre de mes Saldo -> saldo utilizado por los clientes a cierre de mes Genero -> genero del cliente grupo_actividad_eco -> que actividad económica tiene el grupo de clienes Cuidad_res -> ciudad de residencia de los clientes

2.3 Creacion del objeto de analisis temporal indice.ts

2.3.1 Carga de librerias y datasource

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Successfully loaded changepoint package version 2.2.4
##  See NEWS for details of changes.
```

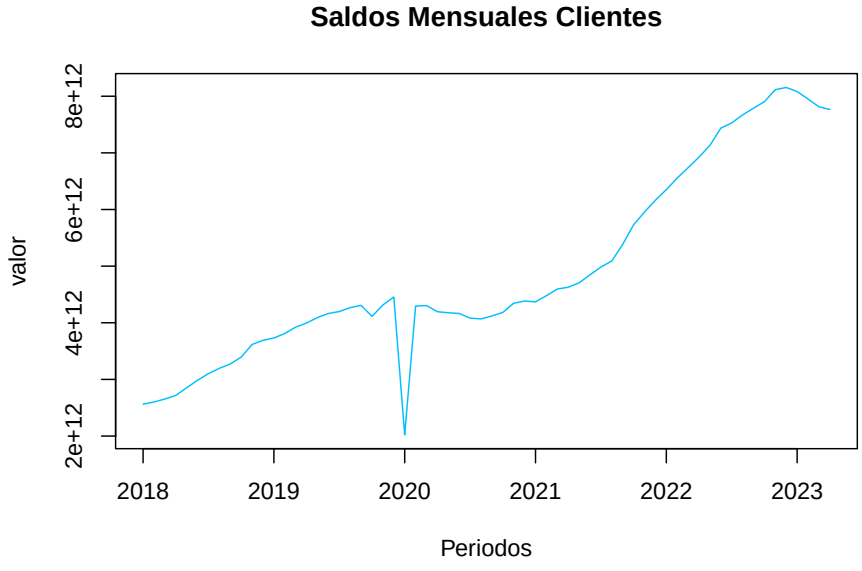
```
## # A tibble: 643,570 x 8
##    Periodo Sub_Tipo N_Clientes DIAS_DE_MORA     Saldo Genero grupo_actividad_eco
##    <chr>   <chr>         <dbl>        <dbl>     <dbl> <chr>  <chr>
##  1 2018-01 CDC               4            0 15824105. Femen~ Dependiente privado
##  2 2018-01 CDC               1            0  6810373. Femen~ Dependiente privado
##  3 2018-01 CDC               6            6 28819502. Femen~ Dependiente privado
##  4 2018-01 CDC              12           63 81343674. Femen~ Dependiente privado
##  5 2018-01 CDC               1           21  7524344. Femen~ Dependiente privado
##  6 2018-01 CDC               4            0 12974213. Femen~ Dependiente privado
##  7 2018-01 CDC               3            1 21348609. Femen~ Dependiente privado
##  8 2018-01 CDC               2            0 11475858. Femen~ Dependiente privado
##  9 2018-01 CDC              10           38 60012355. Femen~ Dependiente privado
## 10 2018-01 CDC               1            0  9034715. Femen~ Dependiente privado
## # i 643,560 more rows
## # i 1 more variable: Cuidad_res <chr>
```

2.3.2 Modificamos el df para que tenga el formato adecuado y lo mostramos

```
##               Jan          Feb          Mar          Apr          May
## 2018 2.562223e+12 2.601532e+12 2.653315e+12 2.717915e+12 2.852608e+12
```

```
## 2019 3.732137e+12 3.810740e+12 3.923380e+12 3.995810e+12 4.092819e+12
## 2020 2.022405e+12 4.295886e+12 4.304185e+12 4.195404e+12 4.177219e+12
## 2021 4.369300e+12 4.478002e+12 4.595047e+12 4.627874e+12 4.705783e+12
## 2022 6.351501e+12 6.555819e+12 6.739823e+12 6.925001e+12 7.132542e+12
## 2023 8.083445e+12 7.951094e+12 7.812947e+12 7.765132e+12
##               Jun           Jul           Aug           Sep           Oct
## 2018 2.986446e+12 3.102493e+12 3.196138e+12 3.272388e+12 3.394038e+12
## 2019 4.164386e+12 4.198177e+12 4.267921e+12 4.307340e+12 4.113506e+12
## 2020 4.164434e+12 4.082507e+12 4.067948e+12 4.120926e+12 4.182877e+12
## 2021 4.846473e+12 4.983882e+12 5.091594e+12 5.385784e+12 5.730839e+12
## 2022 7.435792e+12 7.529328e+12 7.670212e+12 7.789046e+12 7.903472e+12
## 2023
##               Nov           Dec
## 2018 3.617129e+12 3.690229e+12
## 2019 4.314034e+12 4.455499e+12
## 2020 4.344709e+12 4.384188e+12
## 2021 5.955740e+12 6.164705e+12
## 2022 8.115444e+12 8.154174e+12
## 2023
```

Deacuerdo al analisis que deseamos hacer consolidamos nuestra variable de interes saldo agrupandola por periodo de tiempo y procedimos a graficar



2.3.3 Analisis Grafica de serie

Después de graficar la serie de tiempo, podemos observar ciertas características que nos brindan información valiosa. Por ejemplo, en el primer periodo del 2020, podemos notar un pico descendente en el saldo, lo que podría sugerir un posible error en el registro de los datos históricos.

Por otro lado, al examinar el comportamiento general de la serie de tiempo, se observa un aumento en la utilización de los productos a lo largo de los años, especialmente marcado a partir de la mitad del 2020. Este aumento podría deberse a factores como la pandemia y el desempleo, que podrían haber influido en la demanda de estos productos.

2.3.4 Chequeos basicos para confirmar la estructura del contenedor ts

```
## [1] "El tipo de datos del df indice.ts es: "
```

```
## [1] "ts"
```

```
## [1] "La serie de tiempo indice.ts empieza en: "
```

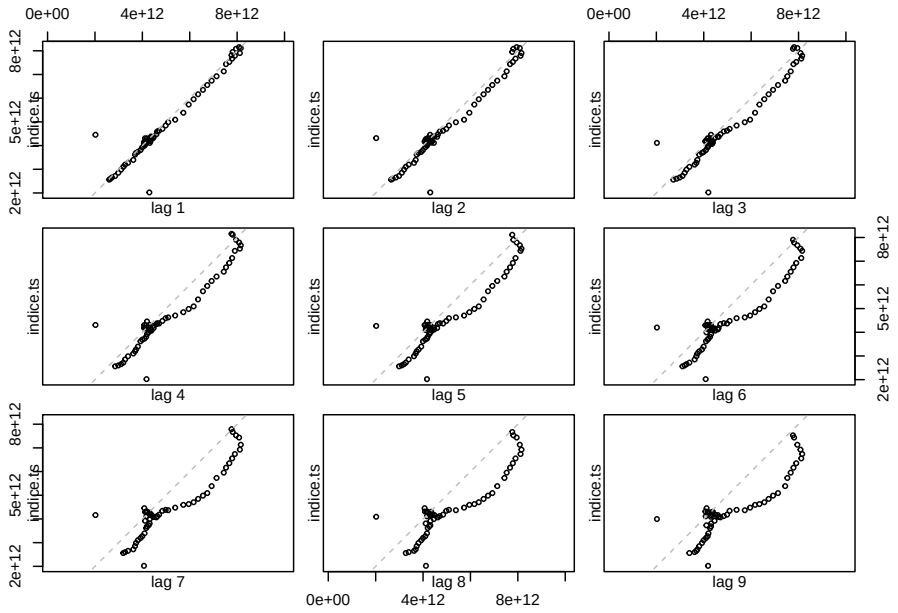
```
## [1] 2018    1
```

```
## [1] "La serie de tiempo indice.ts termina en: "
```

```
## [1] 2023    4
```

2.4 Analisis Descriptivo

2.4.1 Grafica de Rezagos



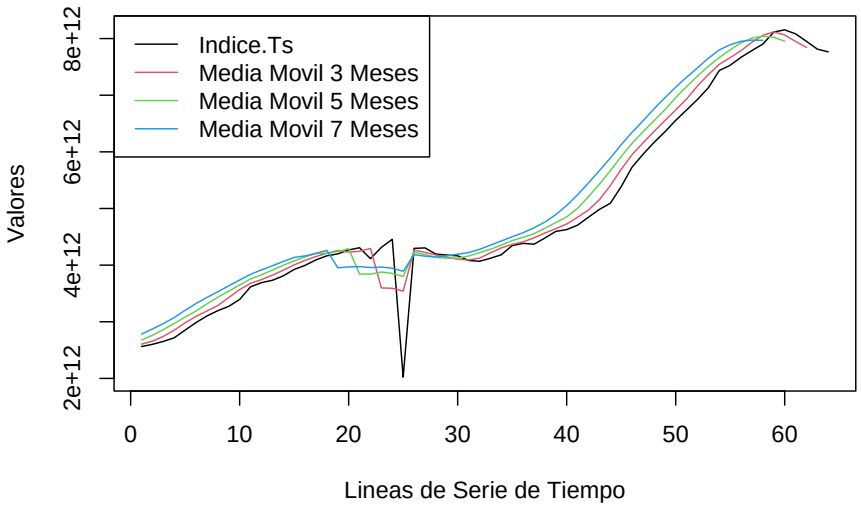
Conclusion: Se observa con claridad que existe una tendencia positiva. Esto sugiere una posible transformacion en una etapa posterior del analisis.

2.4.2 Media Movil

Crearemos a continuacion 3 medias moviles para el objeto ts. Estas tendran 3, 5 y 7 periodos para su calculo.

```
## Media Movil con 3 meses:  2.60569e+12 2.657587e+12 2.741279e+12 2.852323e+12 2.980516e+12 3.09

## Media Movil con 5 meses:  2.677519e+12 2.762363e+12 2.862556e+12 2.97112e+12 3.082015e+12 3.19

## Media Movil con 7 meses:  2.782362e+12 2.872921e+12 2.968758e+12 3.074575e+12 3.203034e+12 3.3
```

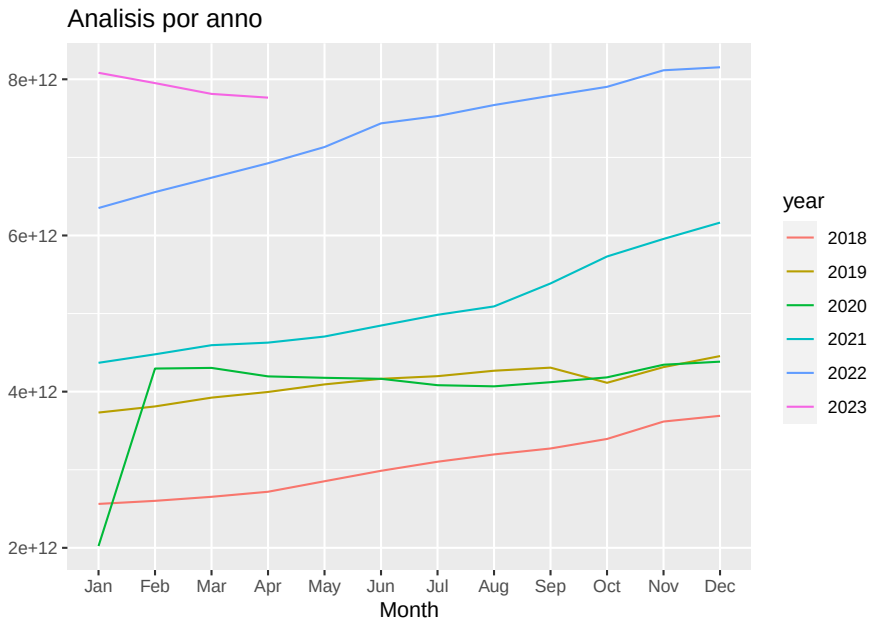
Veamos como es el comportamiento de las mismas en comparacion con los datos originales de la serie de tiempo.


Indice.Ts
Media Movil 3 Meses
Media Movil 5 Meses
Media Movil 7 Meses
Valores
2e+12
4e+12
6e+12
8e+12
0
10
20
30
40
50
60
Lineas de Serie de Tiempo

Chapter 3

Estacionalidad y Descomposicion

3.1 Estacionalidad

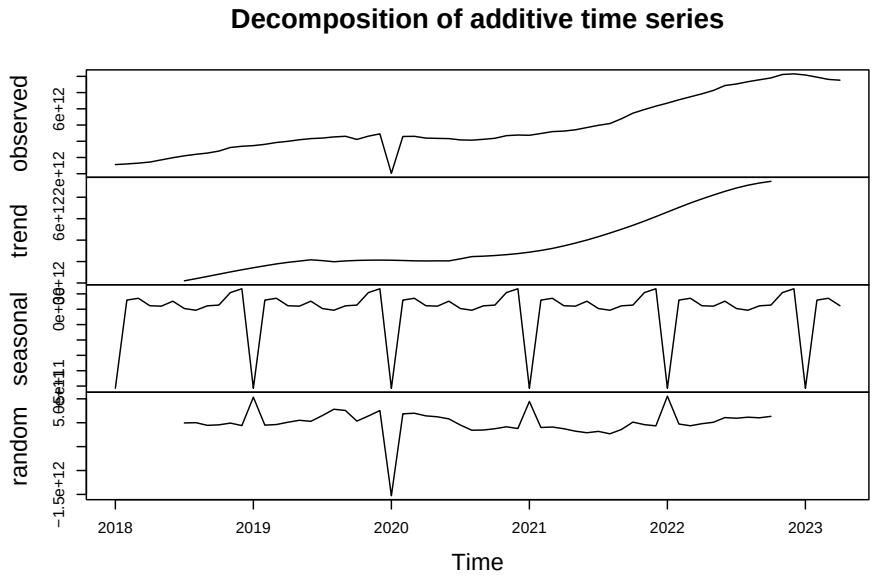


3.1.1 Analisis Inicial

Según el análisis de estacionalidad anual, se observa que la utilización de los productos del banco fue moderada en los años 2018 y 2019, tal como se evidencia en las líneas trazadas para esos años. En el 2020, se observa un pico en la utilización que podría indicar un posible error en la recolección de datos. Posteriormente, se observa una pequeña disminución que posiblemente se debió al inicio de la pandemia y la incertidumbre mundial. A partir de septiembre de 2020, se observa un incremento continuo en la utilización para los años 2021 y 2022, posiblemente como resultado de la duración de la pandemia y la crisis económica global.

En el año 2023, se comienza a evidenciar una disminución en la utilización de los productos del banco, lo que podría deberse al alza de las tasas de interés o a un cambio en el comportamiento de los clientes. Es importante mencionar que se requiere un análisis más detallado para determinar las causas precisas de esta disminución.

3.2 Descomposicion del objeto y analisis



A pesar de presentar un patron recurrente en el componente de estacionalidad, se puede observar un trend en la serie de datos. De igual manera el error no se ve aleatorio sino que por el contrario, presenta un patron constante. Dicho

esto, vamos a comprobar por medio del Augmented Dickery-Fuller test (adf) la estacionalidad del conjunto de datos.

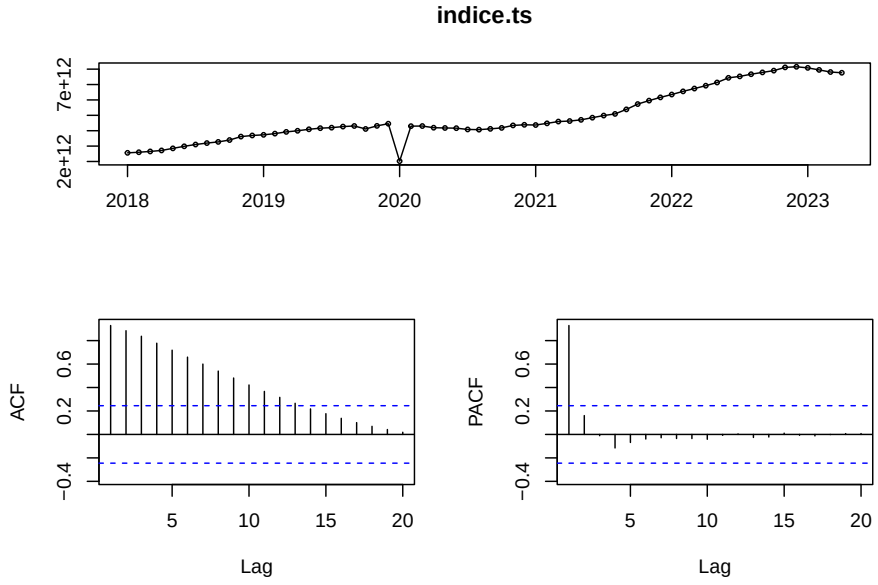
3.2.1 Prueba de Estacionalidad

```
##
##  Augmented Dickey-Fuller Test
##
## data:  indice.ts
## Dickey-Fuller = -1.2017, Lag order = 3, p-value = 0.8984
## alternative hypothesis: stationary
```

De acuerdo al resultado (p-value >0.05), debemos aceptar la H0 la cual nos confirma la no-estacionalidad del conjunto. Esto quiere decir que el objeto indice.ts requerira una transformacion para su posterior procesamiento en el modelo.

3.2.2 Autocorrelacion

Ahora veamos si existe autocorrelacion total o parcial.(acf y pacf tests)



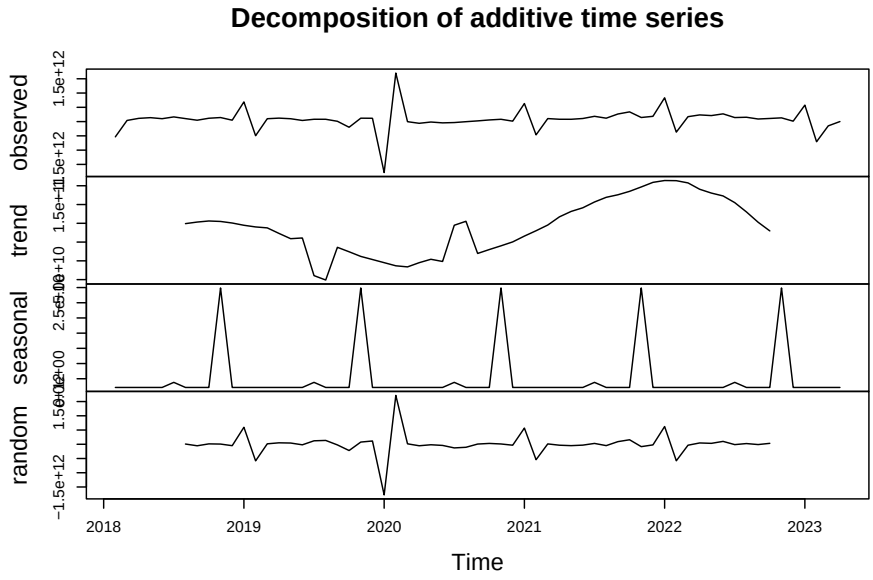
Como se puede observar, existe autocorrelacion entre la variable observada lo cual confirma la tendencia en la serie temporal. Por otro lado no evidenciamos autocorrelacion parcial ya que no encontramos picos por fuera del umbral (0.95)

3.3 Transformacion

```
## [1] "Ajustamos la estacionalidad de la serie de tiempo por medio del comando seasadj
```

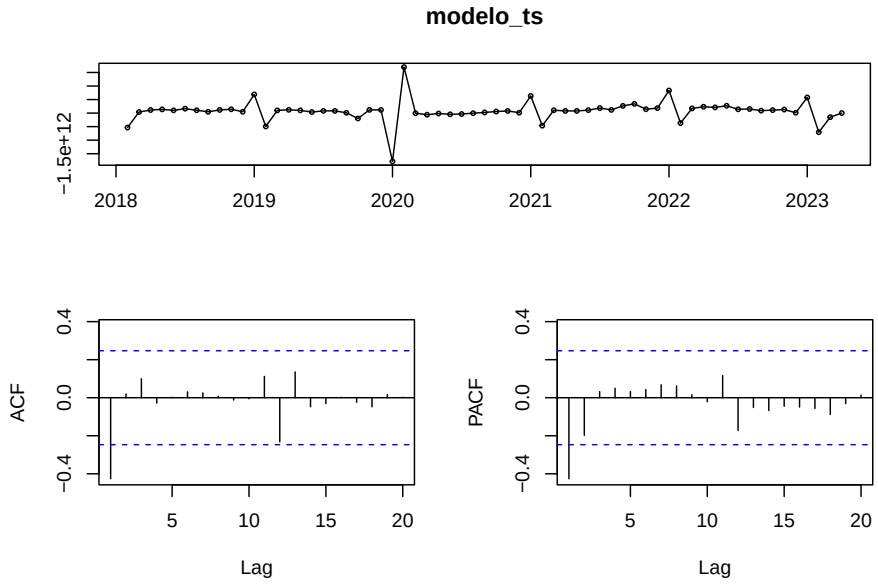
```
## [1] "Luego removemos la tendencia (trend) con el comando diff"
```

```
## [1] "Grafiquemos la nueva serie de tiempo"
```



3.3.1 Validacion de nuestra nueva serie de tiempo transformada

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelo_ts
## Dickey-Fuller = -3.7052, Lag order = 3, p-value = 0.0316
## alternative hypothesis: stationary
```



se observa que la serie es de tipo estacionaria (p-value=0.0316 $<$ 0.05), la varianza y la media son de tipo constante, los datos se mueven alrededor de cero (0).

3.4 Conclusiones

En conclusión, el análisis de serie de tiempo de la utilización de los productos del banco nos ha permitido identificar patrones y tendencias en su utilización.

A partir del análisis de estacionalidad, se observó que la utilización de los productos del banco ha sido moderada en los años 2018 y 2019, y ha aumentado significativamente en el 2020 y 2021. Además, se evidenció una disminución en la utilización para el año 2023.

La descomposición de la serie de tiempo nos permitió identificar las componentes de tendencia, estacionalidad y error, y su análisis nos ha brindado una mejor comprensión de los patrones y comportamientos observados en la utilización de los productos del banco.

Asimismo, se realizó la diferenciación de la serie de tiempo para eliminar la tendencia y hacer la serie estacionaria. Esto nos permitió obtener una serie de tiempo más homogénea, y por lo tanto, una mejor visualización de las fluctuaciones en la utilización de los productos del banco.Como resultado, la nueva serie de tiempo sera nuestra base para la implementacion de modelos de pronostico de los productos observados.

Chapter 4

Holt-Winters(HW) y Suavizamiento Exp.

4.1 HW

Para aplicar la metodología de Holt-Winters y suavizamiento exponencial a la serie de tiempo indice.ts, podemos utilizar las funciones correspondientes de R. La metodología de Holt-Winters es adecuada para modelar series de tiempo con componentes de tendencia y estacionalidad. El resultado del modelo Holt-Winters proporciona las componentes de tendencia (trend), estacionalidad (seasonal) y residuos (residuals). Estas componentes ayudan a comprender los patrones y la estructura de la serie de tiempo.

A continuación, se muestra cómo aplicar ambas técnicas a la serie de tiempo indice.ts:

```
# Aplicar la metodología de Holt-Winters
hw_model <- HoltWinters(indice.ts)

# Obtener las componentes del modelo (tendencia, estacionalidad y residuos)
trend <- hw_model$components$trend
seasonal <- hw_model$components$seasonal
residuals <- hw_model$components$random

# Imprimir las componentes
print("Tendencia:")
```

```
## [1] "Tendencia:"
```

```
print(trend)
```

```
## NULL
```

```
print("Estacionalidad:")
```

```
## [1] "Estacionalidad:"
```

```
print(seasonal)
```

```
## NULL
```

```
print("Residuos:")
```

```
## [1] "Residuos:"
```

```
print(residuals)
```

```
## NULL
```

Suavizamiento exponencial: El suavizamiento exponencial,es útil para suavizar las fluctuaciones en la serie de tiempo.

```
# Aplicar suavizamiento exponencial
smoothed <- HoltWinters(indice.ts, beta = FALSE, gamma = FALSE)$fitted

# Imprimir la serie de tiempo suavizada
print("Serie de tiempo suavizada:")
```

```
## [1] "Serie de tiempo suavizada:"
```

```
print(smoothed)
```

```
##                  xhat        level
## Feb 2018 2.562223e+12 2.562223e+12
## Mar 2018 2.586847e+12 2.586847e+12
## Apr 2018 2.628483e+12 2.628483e+12
## May 2018 2.684505e+12 2.684505e+12
## Jun 2018 2.789807e+12 2.789807e+12
## Jul 2018 2.912985e+12 2.912985e+12
```

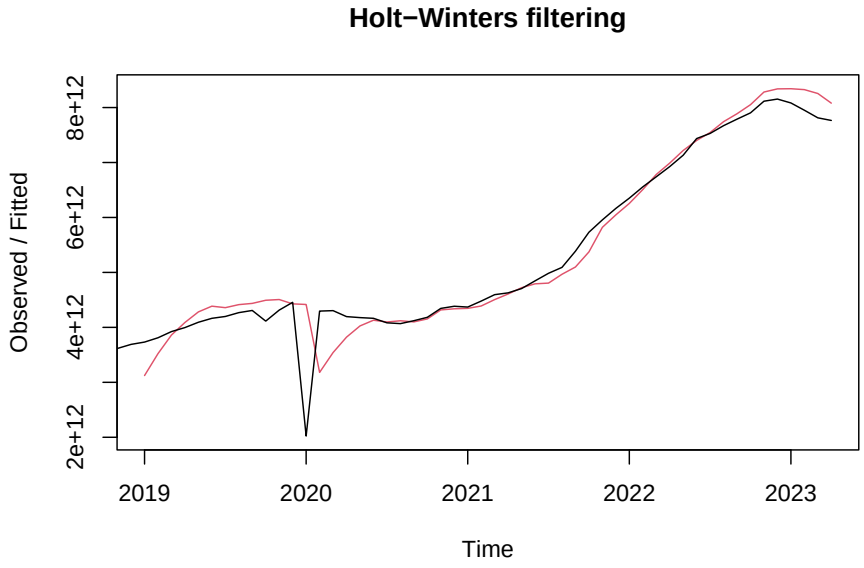
```
## Aug 2018 3.031696e+12 3.031696e+12
## Sep 2018 3.134705e+12 3.134705e+12
## Oct 2018 3.220952e+12 3.220952e+12
## Nov 2018 3.329376e+12 3.329376e+12
## Dec 2018 3.509629e+12 3.509629e+12
## Jan 2019 3.622760e+12 3.622760e+12
## Feb 2019 3.691275e+12 3.691275e+12
## Mar 2019 3.766110e+12 3.766110e+12
## Apr 2019 3.864626e+12 3.864626e+12
## May 2019 3.946802e+12 3.946802e+12
## Jun 2019 4.038270e+12 4.038270e+12
## Jul 2019 4.117271e+12 4.117271e+12
## Aug 2019 4.167952e+12 4.167952e+12
## Sep 2019 4.230574e+12 4.230574e+12
## Oct 2019 4.278661e+12 4.278661e+12
## Nov 2019 4.175205e+12 4.175205e+12
## Dec 2019 4.262169e+12 4.262169e+12
## Jan 2020 4.383274e+12 4.383274e+12
## Feb 2020 2.904388e+12 2.904388e+12
## Mar 2020 3.776045e+12 3.776045e+12
## Apr 2020 4.106880e+12 4.106880e+12
## May 2020 4.162333e+12 4.162333e+12
## Jun 2020 4.171658e+12 4.171658e+12
## Jul 2020 4.167133e+12 4.167133e+12
## Aug 2020 4.114122e+12 4.114122e+12
## Sep 2020 4.085198e+12 4.085198e+12
## Oct 2020 4.107579e+12 4.107579e+12
## Nov 2020 4.154747e+12 4.154747e+12
## Dec 2020 4.273742e+12 4.273742e+12
## Jan 2021 4.342927e+12 4.342927e+12
## Feb 2021 4.359448e+12 4.359448e+12
## Mar 2021 4.433712e+12 4.433712e+12
## Apr 2021 4.534775e+12 4.534775e+12
## May 2021 4.593094e+12 4.593094e+12
## Jun 2021 4.663684e+12 4.663684e+12
## Jul 2021 4.778186e+12 4.778186e+12
## Aug 2021 4.907038e+12 4.907038e+12
## Sep 2021 5.022646e+12 5.022646e+12
## Oct 2021 5.250122e+12 5.250122e+12
## Nov 2021 5.551251e+12 5.551251e+12
## Dec 2021 5.804630e+12 5.804630e+12
## Jan 2022 6.030187e+12 6.030187e+12
## Feb 2022 6.231463e+12 6.231463e+12
## Mar 2022 6.434645e+12 6.434645e+12
## Apr 2022 6.625813e+12 6.625813e+12
## May 2022 6.813229e+12 6.813229e+12
```

```
## Jun 2022 7.013252e+12 7.013252e+12
## Jul 2022 7.277938e+12 7.277938e+12
## Aug 2022 7.435413e+12 7.435413e+12
## Sep 2022 7.582495e+12 7.582495e+12
## Oct 2022 7.711882e+12 7.711882e+12
## Nov 2022 7.831897e+12 7.831897e+12
## Dec 2022 8.009515e+12 8.009515e+12
## Jan 2023 8.100132e+12 8.100132e+12
## Feb 2023 8.089679e+12 8.089679e+12
## Mar 2023 8.002867e+12 8.002867e+12
## Apr 2023 7.883898e+12 7.883898e+12
```

La función HoltWinters con los argumentos beta = FALSE y gamma = FALSE aplica el suavizamiento exponencial simple sin considerar la componente de estacionalidad en el modelo.

Al aplicar esta metodología y el suavizamiento exponencial a la serie de tiempo indice.ts, se obtendrá la tendencia, la estacionalidad, los residuos y la serie de tiempo suavizada. Estos resultados ayudarán a comprender la estructura y los patrones de la serie de tiempo.

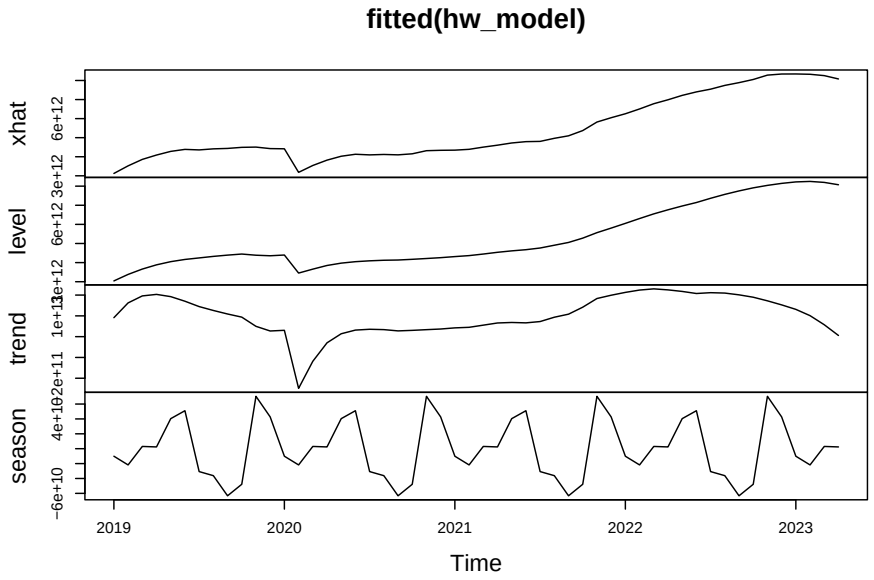
```
hw_model=HoltWinters(indice.ts, seasonal = "additive")
plot(hw_model)
```



Utilizando el comando HoltWinters en R, podemos generar una gráfica en color rojo que representa una serie de datos aproximada a los datos originales en

color negro. Cabe destacar que toda serie de tiempo consta de un componente constante, tendencia y estacionalidad. Para ajustar la serie según estas características, realizamos lo siguiente:

```
plot(fitted(hw_model))
```



En la gráfica se puede observar la descomposición en las cuatro componentes mencionadas previamente.

El método Holt Winters nos permite realizar predicciones utilizando la serie de tiempo. A continuación, se muestra el proceso de generación de predicciones.

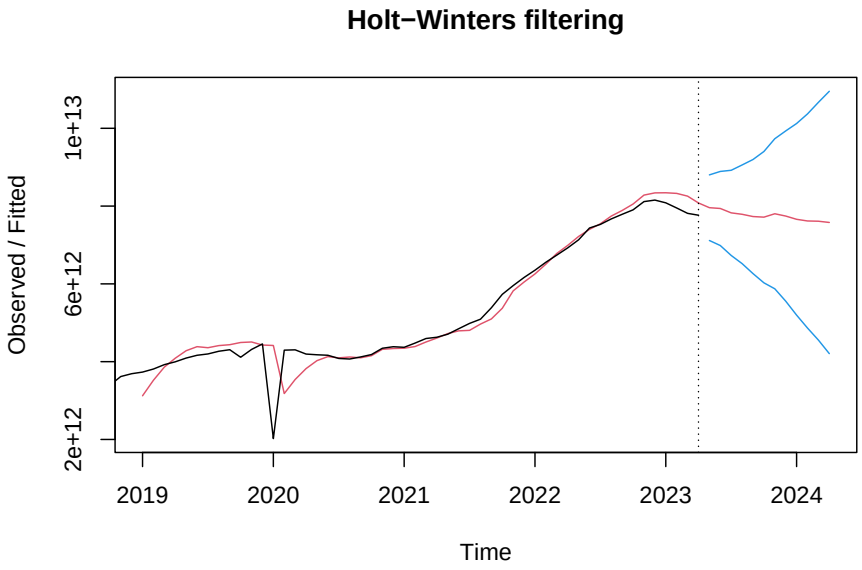
```
pred=predict(hw_model, 12, prediction.interval = TRUE)
pred
```

```
##                    fit          upr          lwr
## May 2023 7.958686e+12 8.802116e+12 7.115255e+12
## Jun 2023 7.938280e+12 8.890571e+12 6.985989e+12
## Jul 2023 7.825482e+12 8.920472e+12 6.730492e+12
## Aug 2023 7.788949e+12 9.056689e+12 6.521209e+12
## Sep 2023 7.730847e+12 9.197373e+12 6.264322e+12
## Oct 2023 7.715284e+12 9.403168e+12 6.027400e+12
## Nov 2023 7.802406e+12 9.731469e+12 5.873343e+12
## Dec 2023 7.743702e+12 9.931634e+12 5.555770e+12
## Jan 2024 7.659827e+12 1.012267e+13 5.196985e+12
```

```
## Feb 2024 7.617072e+12 1.036958e+13 4.864568e+12
## Mar 2024 7.610883e+12 1.066677e+13 4.554994e+12
## Apr 2024 7.579164e+12 1.095133e+13 4.207002e+12
```

Se generan predicciones para los próximos 12 meses (May 2023 - Apr 2024) y se procede a graficar la predicción.

```
plot(hw_model, pred)
```



En la gráfica se muestran la tendencia de los valores pronosticados junto con sus intervalos de confianza.

Tambien podemos utilizar la funcion hw

```
# Calcular el modelo de Holt-Winters utilizando la función hw():

hw_model1 <- HoltWinters(indice.ts)
hw_model1
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = indice.ts)
##
## Smoothing parameters:
```

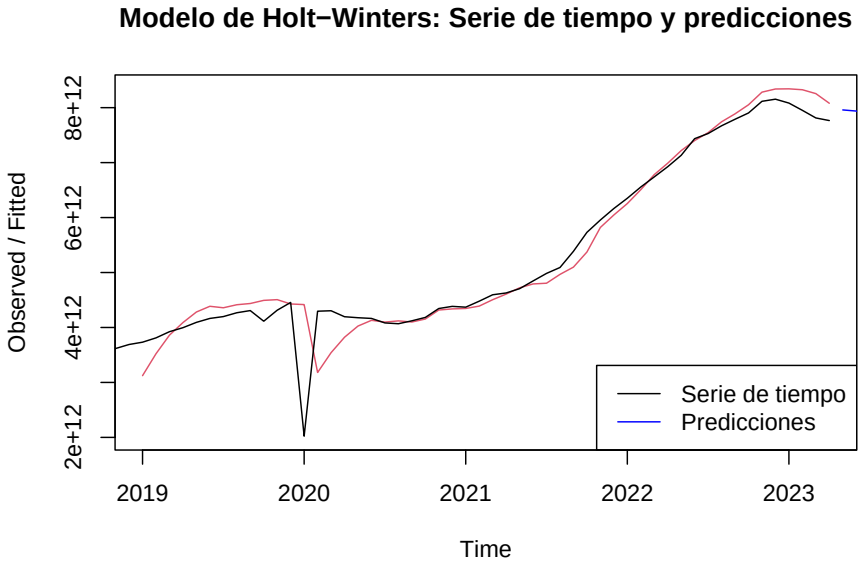
```
##  alpha: 0.407578
##  beta : 0.2861578
##  gamma: 0
##
## Coefficients:
##              [,1]
## a   7949360142242
## b    -31047949847
## s1    40373454460
## s2    51015741884
## s3   -30734639955
## s4   -36219767637
## s5   -63272955789
## s6   -47788675079
## s7    70381305327
## s8    42725269292
## s9   -10101155804
## s10  -21808809060
## s11    3050492943
## s12    2379739417
```

se generan predicciones para los próximos 12 periodos.

```
# Obtener las predicciones del modelo para un horizonte de tiempo determinado:
library(forecast)
predictions <- forecast(hw_model1, h = 12)  # Predicciones para los próximos 12 periodos
```

Esto generará un gráfico que muestra la serie de tiempo original y las predicciones del modelo de Holt-Winters.

```
plot(hw_model1, main = "Modelo de Holt-Winters: Serie de tiempo y predicciones")
lines(predictions$mean, col = "blue")
legend("bottomright", legend = c("Serie de tiempo", "Predicciones"), col = c("black", "blue"), lt
```


Modelo de Holt−Winters: Serie de tiempo y predicciones
Observed / Fitted
2e+12
4e+12
6e+12
8e+12
2019
2020
2021
2022
2023
Time
Serie de tiempo
Predicciones

Chapter 5

Modelo ARIMA

##Metodología Box-Jenkins para identificar modelos autoregresivos integrados de media móvil (ARIMA) para analizar y predecir valores futuros de serie de tiempo.

En esta seccion, intentaremos abordar el algoritmo ARIMA dentro de la serie de tiempo. Primero consrtruiremos el modelo optimo AR, luego MA y posteriormente utilizaremos la funcion Autoarima para encontrar los parametros optimos.

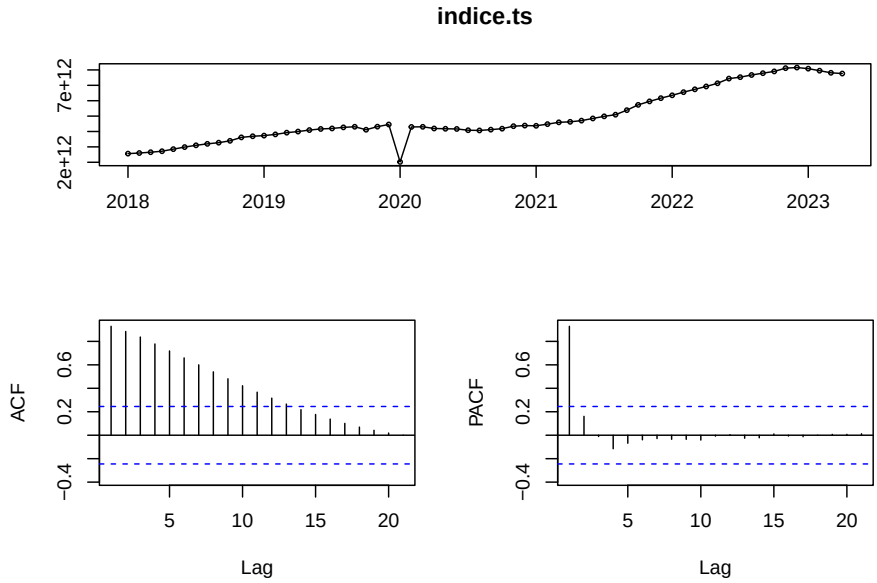
5.1 Modelos

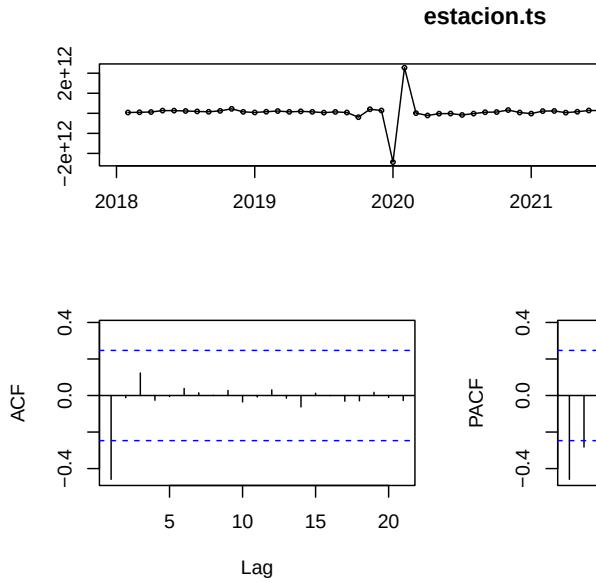
```
## -- Attaching packages --------------------------------------------- fpp2 2.5 --

## v fma       2.5     v expsmooth 2.3

##

## [1] "Verificamos la estacionalidad del modelo (p<0.05)"

##
##  Augmented Dickey-Fuller Test
##
## data:  modelo_ts
## Dickey-Fuller = -3.7052, Lag order = 3, p-value = 0.0316
## alternative hypothesis: stationary
```

Como se puede observar, se rechaza la hipotesis ho y aceptamos la alterna. xxxxxxx , dentro de nuestros modelos ARIMA podemos asegurar que el parametro d = 0.

5.1.1 Modelo basado solamente en Auro Regresion (AR). Debemos ubicar los parametros d y q en 0.

Por medio del analisis ACF y PACF verificamos los lags





Procedemos a diferenciarla ya que es No-Estacionaria

```
## [1] 0
```

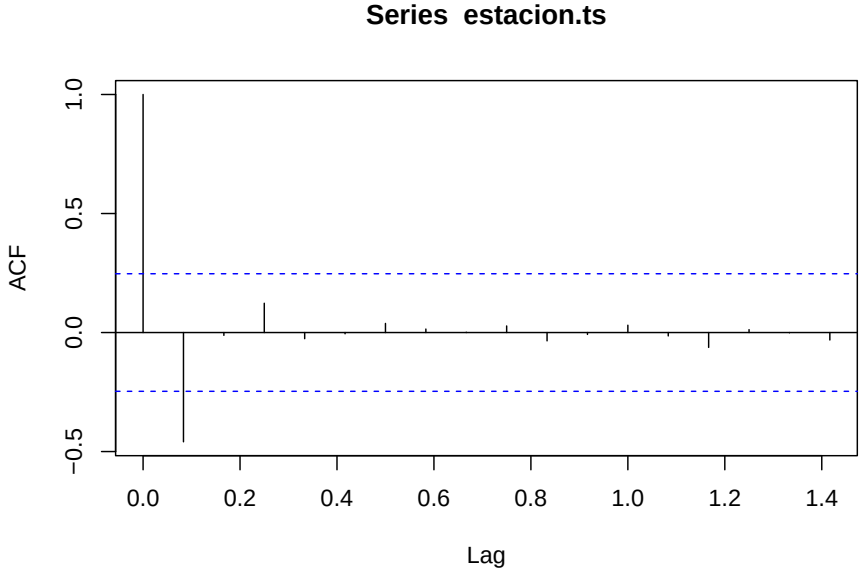
En el grafico PAFC podemos ver los lag en 1 como punto significativo de cambio.

Construyamos entonces nuestro modelo con valor AR (p) = 1. A tener en cuenta: Al tener un modelo diferenciado, debemos especificar que no se incluya la media en los calculos ya que la misma es 0.

```
##
## Call:
## arima(x = estacion.ts, order = c(1, 0, 0), include.mean = F)
##
## Coefficients:
##           ar1
##       -0.4010
## s.e.   0.1139
##
## sigma^2 estimated as 1.621e+23:  log likelihood = -1772.91,  aic = 3549.83
```

5.1.2 Modelo basado solamente en el Moving Average (MA). Parametros p y d en 0

Utilizando el grafico ACF revisamos cual seria el punto de inflexion y luego procedemos a crear nuestro modelo



```
## [1] "Observamos que el 2do lag contiene el ultimo cambio significativo. Ademas el 1e
```

```
##
## Call:
## arima(x = estacion.ts, order = c(0, 0, 2), include.mean = F)
##
## Coefficients:
##           ma1     ma2
##       -0.5263  0.2126
## s.e.   0.1412  0.1260
##
## sigma^2 estimated as 1.556e+23:  log likelihood = -1771.7,  aic = 3549.39
```

5.2 Modelo ARIMA. Validacion por medio de la funcion auto.arima

Como pudimos comprobar en los puntos anteriores, la seleccion de parametros del modelo clasico Arima, depende de las caracteristicas de la serie de tiempo a evaluar. Es por ello que decidimos comparar el calculo manual de nuestras variables con respecto al modelo automatico que viene incluido en la libreria de forecast auto.arima.

Para mostrar los resultados, debemos habilitar la opcion trace la cual permite evaluar todos los modelos que pudiesen resultar de la serie de tiempo. Asi mismo, decidimos utilizar 2 parametros mas y configurarlos en Falso - Stepwise y Approximation - los cuales maximizan la busqueda del mejor modelo, al tiempo que sacrifica tanto numero de modelos a evaluar asi como velocidad de respuesta.

```
##
##  ARIMA(0,0,0)            with zero mean     : 3559.137
##  ARIMA(0,0,0)(0,0,1)[12] with zero mean     : 3561.092
##  ARIMA(0,0,0)(1,0,0)[12] with zero mean     : 3561.089
##  ARIMA(0,0,0)(1,0,1)[12] with zero mean     : 3563.321
##  ARIMA(0,0,1)            with zero mean     : 3550.238
##  ARIMA(0,0,1)(0,0,1)[12] with zero mean     : 3551.857
##  ARIMA(0,0,1)(1,0,0)[12] with zero mean     : 3551.834
##  ARIMA(0,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,2)            with zero mean     : 3549.801
##  ARIMA(0,0,2)(0,0,1)[12] with zero mean     : 3551.52
##  ARIMA(0,0,2)(1,0,0)[12] with zero mean     : 3551.505
##  ARIMA(0,0,2)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(0,0,3)            with zero mean     : 3549.69
##  ARIMA(0,0,3)(0,0,1)[12] with zero mean     : 3551.74
##  ARIMA(0,0,3)(1,0,0)[12] with zero mean     : 3551.735
```

```
##  ARIMA(0,0,3)(1,0,1)[12] with zero mean     : 3554.18
##  ARIMA(0,0,4)            with zero mean     : 3551.745
##  ARIMA(0,0,4)(0,0,1)[12] with zero mean     : 3553.955
##  ARIMA(0,0,4)(1,0,0)[12] with zero mean     : 3553.952
##  ARIMA(0,0,5)            with zero mean     : 3553.08
##  ARIMA(1,0,0)            with zero mean     : 3550.027
##  ARIMA(1,0,0)(0,0,1)[12] with zero mean     : 3551.701
##  ARIMA(1,0,0)(1,0,0)[12] with zero mean     : 3551.685
##  ARIMA(1,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,1)            with zero mean     : 3551.278
##  ARIMA(1,0,1)(0,0,1)[12] with zero mean     : 3552.931
##  ARIMA(1,0,1)(1,0,0)[12] with zero mean     : 3552.909
##  ARIMA(1,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(1,0,2)            with zero mean     : 3544.017
##  ARIMA(1,0,2)(0,0,1)[12] with zero mean     : 3546.351
##  ARIMA(1,0,2)(1,0,0)[12] with zero mean     : 3546.354
##  ARIMA(1,0,2)(1,0,1)[12] with zero mean     : 3548.724
##  ARIMA(1,0,3)            with zero mean     : 3546.198
##  ARIMA(1,0,3)(0,0,1)[12] with zero mean     : 3548.603
##  ARIMA(1,0,3)(1,0,0)[12] with zero mean     : 3548.607
##  ARIMA(1,0,4)            with zero mean     : 3548.636
##  ARIMA(2,0,0)            with zero mean     : 3550.466
##  ARIMA(2,0,0)(0,0,1)[12] with zero mean     : 3552.099
##  ARIMA(2,0,0)(1,0,0)[12] with zero mean     : 3552.075
##  ARIMA(2,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(2,0,1)            with zero mean     : 3552.408
##  ARIMA(2,0,1)(0,0,1)[12] with zero mean     : 3554.165
##  ARIMA(2,0,1)(1,0,0)[12] with zero mean     : 3554.143
##  ARIMA(2,0,1)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(2,0,2)            with zero mean     : 3546.184
##  ARIMA(2,0,2)(0,0,1)[12] with zero mean     : 3548.584
##  ARIMA(2,0,2)(1,0,0)[12] with zero mean     : 3548.589
##  ARIMA(2,0,3)            with zero mean     : 3548.597
##  ARIMA(3,0,0)            with zero mean     : 3551.834
##  ARIMA(3,0,0)(0,0,1)[12] with zero mean     : 3553.7
##  ARIMA(3,0,0)(1,0,0)[12] with zero mean     : 3553.683
##  ARIMA(3,0,0)(1,0,1)[12] with zero mean     : Inf
##  ARIMA(3,0,1)            with zero mean     : 3548.871
##  ARIMA(3,0,1)(0,0,1)[12] with zero mean     : Inf
##  ARIMA(3,0,1)(1,0,0)[12] with zero mean     : 3551.285
##  ARIMA(3,0,2)            with zero mean     : 3548.575
##  ARIMA(4,0,0)            with zero mean     : 3552.398
##  ARIMA(4,0,0)(0,0,1)[12] with zero mean     : 3554.534
##  ARIMA(4,0,0)(1,0,0)[12] with zero mean     : 3554.529
##  ARIMA(4,0,1)            with zero mean     : 3550.555
##  ARIMA(5,0,0)            with zero mean     : 3553.42
```

```
##
##
##
##  Best model: ARIMA(1,0,2)            with zero mean
```

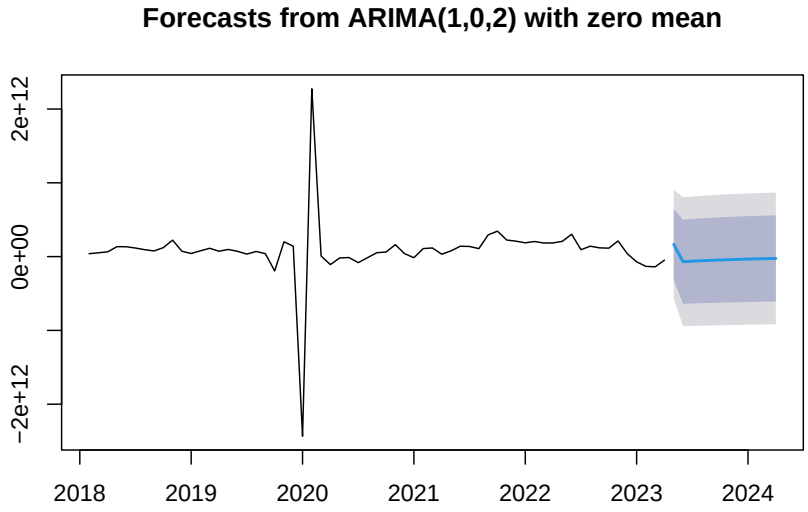
```
## Series: estacion.ts
## ARIMA(1,0,2) with zero mean
##
## Coefficients:
##          ar1      ma1     ma2
##       0.9023  -1.5331  0.6856
## s.e.  0.0701   0.1188  0.1099
##
## sigma^2 = 1.417e+23:  log likelihood = -1767.66
## AIC=3543.33   AICc=3544.02   BIC=3551.9
```

Como resultado, podemos concluir que el Modelo ARIMA mas optimo para nuestea serie de datos es el que utiliza un AR = 1, MA = 2 y 0 en su atributo diferenciador.

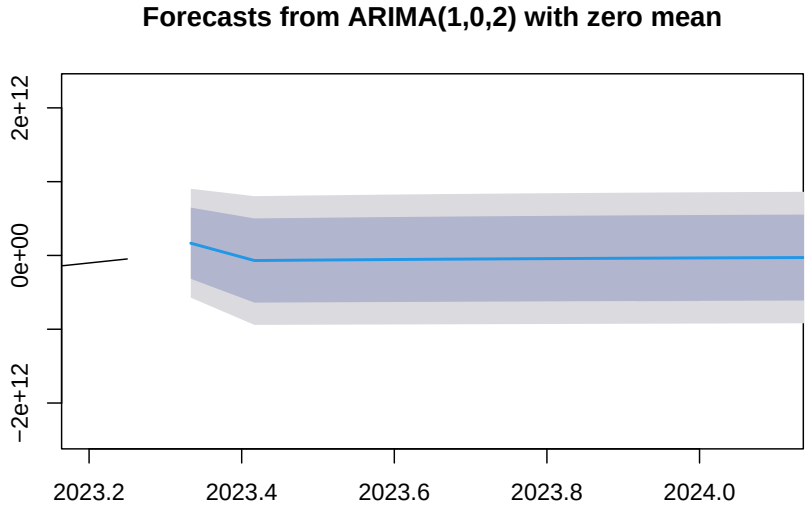
5.3 Analisis

5.3.1 Prediccion del Modelo

Utilicemos nuestro modelo ARIMA para pronosticar los siguintes 12 meses de saldo en las cuentas del banco.




```
## [1] "Veamos mas en detalle la prediccion de los valores"
```




```
## [1] "Valores de la prediccion:"
```

```
##                Jan           Feb           Mar           Apr           May
## 2023                                                        165668225906
## 2024 -33590498111 -30309584374 -27349130158 -24677834944
##                Jun           Jul           Aug           Sep           Oct
## 2023 -68971819772 -62235075641 -56156335340 -50671329091 -45722064595
## 2024
##                Nov           Dec
## 2023 -41256213885 -37226560069
## 2024
```

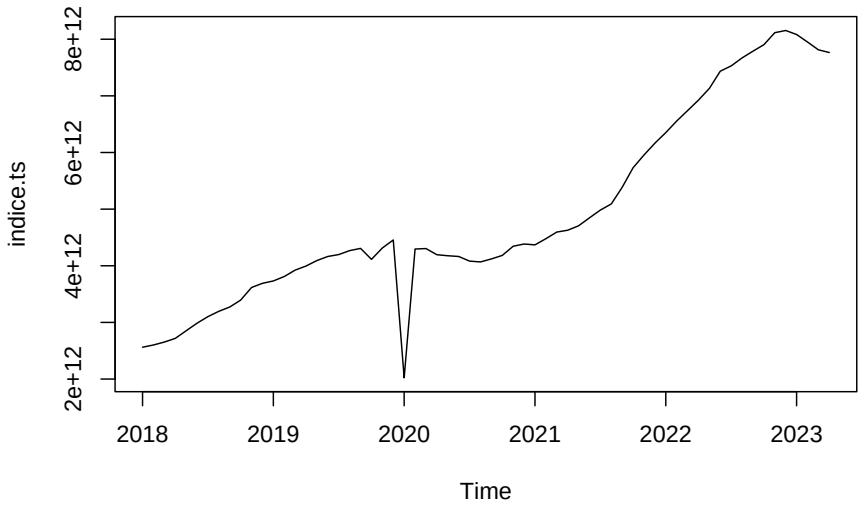
Ahora hagamos las validaciones del modelo

```
## [1] "Error in solve.default(res$hessian * n.used, A) :\nLapack routine dgesv: syste
```

```
## [1] "Al aplicar la funcion tso para encontrar los outliers, nos arroja un error de
```

5.3.2 Diferenciacion Logaritmica

```
# Genero mi objeto ts para el analisis
indice.ts <- ts(datos$Saldo, start = c(2018,1), frequency = 12)
plot(indice.ts)
```



5.3.2.1 Diferenciacion por Logaritmo

```
mits <- log(indice.ts)

# Prueba de Estacionariedad
adf.test(mits)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mits
## Dickey-Fuller = -1.9544, Lag order = 3, p-value = 0.5934
## alternative hypothesis: stationary
```

```
print("Continua siendo No-Estacionaria. Procederemos a diferenciarla")
```

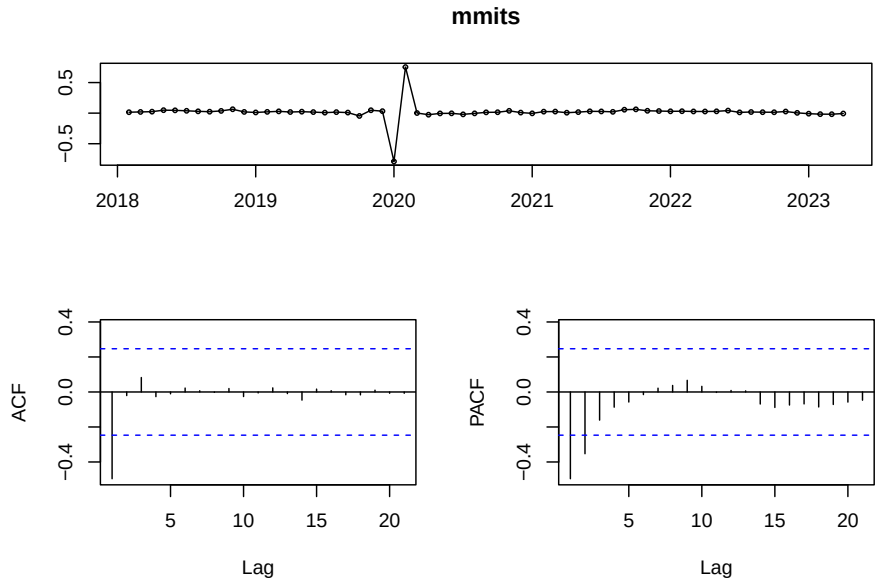
```
## [1] "Continua siendo No-Estacionaria. Procederemos a diferenciarla"
```

```
mmits <- diff(mits)
adf.test(mmits)
```

```
## Warning in adf.test(mmits): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mmits
## Dickey-Fuller = -5.3844, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
tsdisplay(mmits)
```




```
adf.test(mmits)
```

```
## Warning in adf.test(mmits): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  mmits
## Dickey-Fuller = -5.3844, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

Finalmente podemos recharar la h0 ya que p<0.05

5.3.2.2 Procedamos a calular el modelo Arima mas optimo:

```
mimod <- auto.arima(mmits, trace = T)
```

```
##
##  ARIMA(2,0,2)(1,0,1)[12] with non-zero mean : Inf
##  ARIMA(0,0,0)            with non-zero mean : -65.53048
##  ARIMA(1,0,0)(1,0,0)[12] with non-zero mean : -78.58046
```

```
##  ARIMA(0,0,1)(0,0,1)[12] with non-zero mean : -88.02136
##  ARIMA(0,0,0)            with zero mean     : -66.66478
##  ARIMA(0,0,1)            with non-zero mean : -90.27983
##  ARIMA(0,0,1)(1,0,0)[12] with non-zero mean : -88.01915
##  ARIMA(0,0,1)(1,0,1)[12] with non-zero mean : Inf
##  ARIMA(1,0,1)            with non-zero mean : -88.46913
##  ARIMA(0,0,2)            with non-zero mean : -88.62053
##  ARIMA(1,0,0)            with non-zero mean : -80.81767
##  ARIMA(1,0,2)            with non-zero mean : -86.98859
##  ARIMA(0,0,1)            with zero mean     : -83.77733
##
##  Best model: ARIMA(0,0,1)            with non-zero mean
```

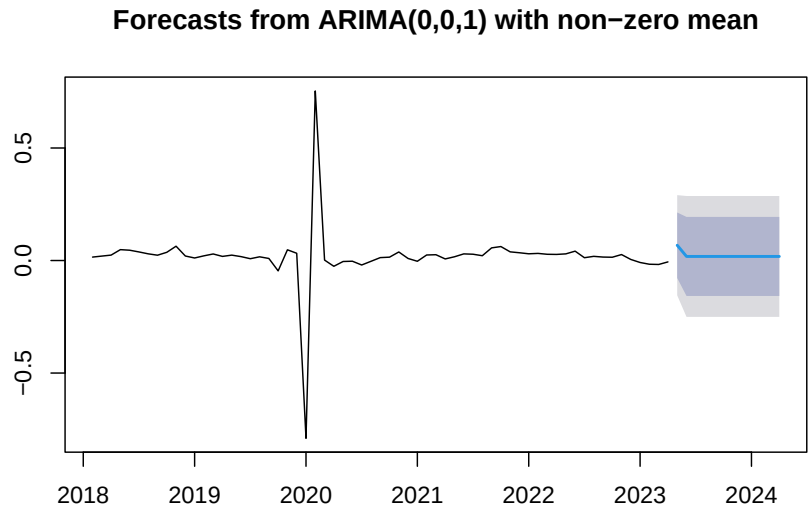
```
mimod
```

```
## Series: mmits
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##           ma1    mean
##       -0.6742  0.0182
## s.e.   0.0846  0.0047
##
## sigma^2 = 0.01291:  log likelihood = 48.34
## AIC=-90.69   AICc=-90.28   BIC=-84.26
```

```
# Hagamos la prediccion de los siguientes 12 meses.
mifore <- forecast(mimod, h=12)
mifore
```

```
##          Point Forecast       Lo 80     Hi 80      Lo 95     Hi 95
## May 2023     0.06780752 -0.07779118 0.2134062 -0.1548665 0.2904815
## Jun 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Jul 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Aug 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Sep 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Oct 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Nov 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Dec 2023     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Jan 2024     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Feb 2024     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Mar 2024     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
## Apr 2024     0.01815148 -0.15744629 0.1937492 -0.2504021 0.2867051
```

```
plot(mifore)
```



5.3.3 Revisemos el efecto de los outliers:

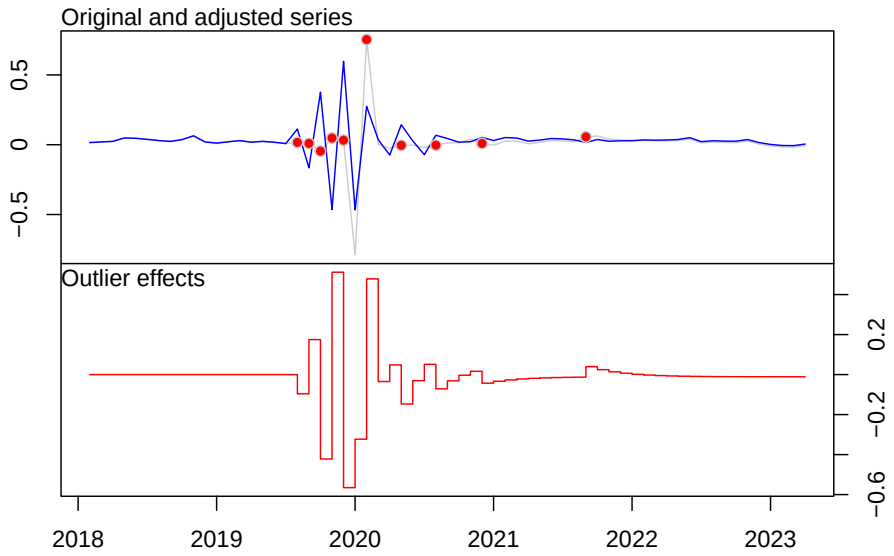
```
outliers_excess_ts <- tso(mmits)
```

```
## Warning in locate.outliers.iloop(resid = resid, pars = pars, cval = cval, :
## stopped when 'maxit.iloop' was reached

## Warning in locate.outliers.iloop(resid = resid, pars = pars, cval = cval, :
## stopped when 'maxit.iloop' was reached

## Warning in locate.outliers.iloop(resid = resid, pars = pars, cval = cval, :
## stopped when 'maxit.iloop' was reached

## Warning in locate.outliers.iloop(resid = resid, pars = pars, cval = cval, :
## stopped when 'maxit.iloop' was reached


## Warning in locate.outliers.oloop(y = y, fit = fit, types = types, cval = cval,
## : stopped when 'maxit.oloop = 4' was reached
```

```
outliers_excess_ts
```

```
## Series: mmits
## Regression with ARIMA(2,0,0) errors
##
## Coefficients:
##           ar1      ar2  intercept     TC19    LS20     TC21    AO22     TC23
##       -1.4393  -0.6456     0.0292  -0.0961  0.2422  -0.6172  0.7346  -0.4820
## s.e.   0.0981   0.0953     0.0038   0.0226  0.0286   0.0719  0.0863   0.0285
##         AO25     TC28    LS31     LS35    TC44
##       0.6319  -0.2536  -0.180  -0.0733  0.0521
## s.e.  0.0492   0.0384   0.022   0.0130  0.0145
##
## sigma^2 = 0.002827:  log likelihood = 101.48
## AIC=-174.96   AICc=-166.21   BIC=-144.96
##
## Outliers:
##    type ind    time  coefhat   tstat
## 1    TC  19 2019:08 -0.09610  -4.262
## 2    LS  20 2019:09  0.24224   8.461
## 3    TC  21 2019:10 -0.61716  -8.580
## 4    AO  22 2019:11  0.73456   8.516
## 5    TC  23 2019:12 -0.48196 -16.888
## 6    AO  25 2020:02  0.63191  12.842
## 7    TC  28 2020:05 -0.25360  -6.606
## 8    LS  31 2020:08 -0.17998  -8.190
## 9    LS  35 2020:12 -0.07327  -5.620
## 10   TC  44 2021:09  0.05214   3.608
```

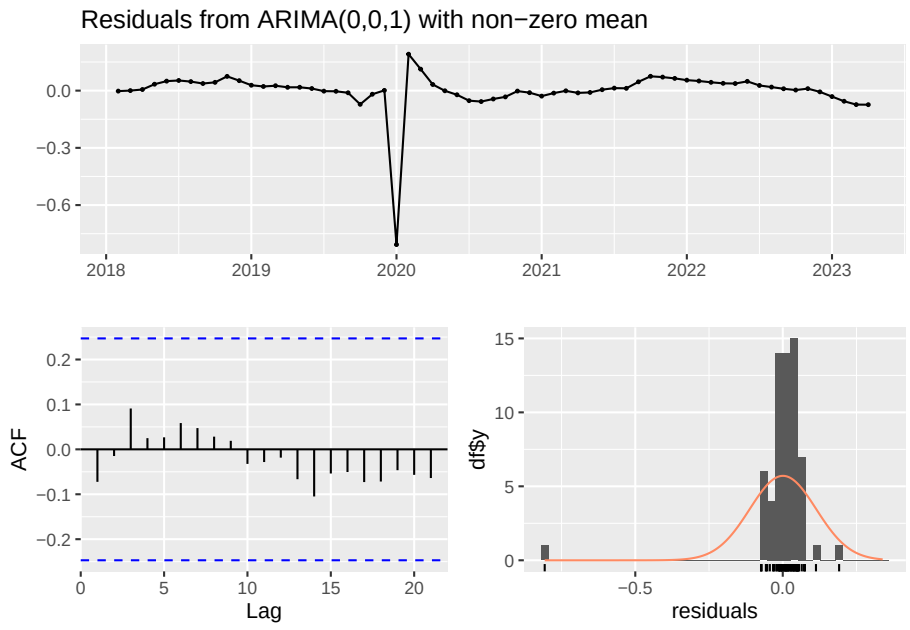
```
plot(outliers_excess_ts)
```



La gráfica de Outliers muestra los 10 valores que difieren significativamente del patrón general de la serie de tiempo.

5.3.3.1 Por ultimo haremos un check de los residuos.

```
checkresiduals(mimod)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 2.0482, df = 12, p-value = 0.9993
##
## Model df: 1.   Total lags used: 13
```

```
residuales <- mimod$residuals

t.test(residuales, alternative='two.sided', conf.level=0.95, mu=0)
```

```
##
##  One Sample t-test
##
## data:  residuales
## t = 0.047608, df = 62, p-value = 0.9622
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.02770451  0.02905634
## sample estimates:
##    mean of x
## 0.0006759195
```

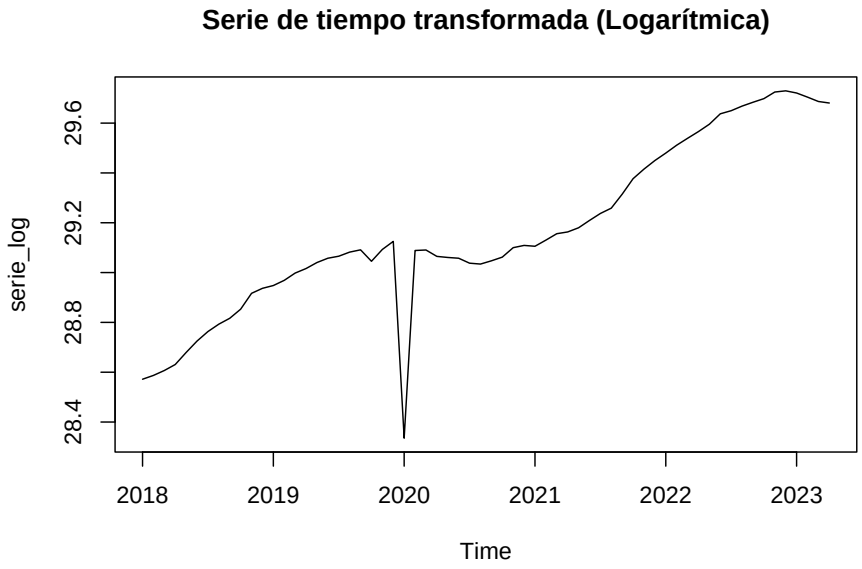
Con esta grafica podemos concluir que los residuales presentan una distribucion normal, que los errores residuales se mantienen dentro del rango de significancia aceptable (0.95) y por medio de un t-test pudimos corroborar que la media de los residuos es 0.

Chapter 6

Modelos Logaritmicos, Prophet y otros

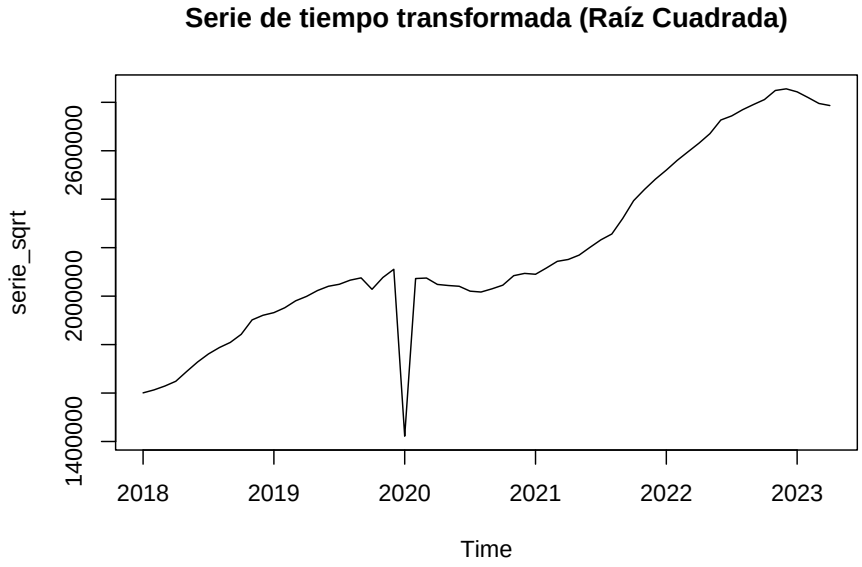
Transformación logarítmica: La transformación logarítmica es útil cuando hay una tendencia exponencial en la serie de tiempo y la varianza aumenta con el nivel de la serie. Aplicar la transformación logarítmica puede ayudar a reducir la tendencia y estabilizar la varianza.se Puede utilizar la función log() para aplicar esta transformación:

```
# Aplicar la transformación logarítmica
serie_log <- log(indice.ts)

# Graficar la serie transformada
plot(serie_log, main = "Serie de tiempo transformada (Logarítmica)")
```



Transformación de raíz cuadrada: La transformación de raíz cuadrada se utiliza cuando la varianza aumenta con el nivel de la serie de tiempo. Al aplicar esta transformación, se reduce la dispersión de los valores más altos y se estabiliza la varianza. Puedes utilizar la función sqrt() para aplicar esta transformación:

```
# Aplicar la transformación de raíz cuadrada
serie_sqrt <- sqrt(indice.ts)

# Graficar la serie transformada
plot(serie_sqrt, main = "Serie de tiempo transformada (Raíz Cuadrada)")
```

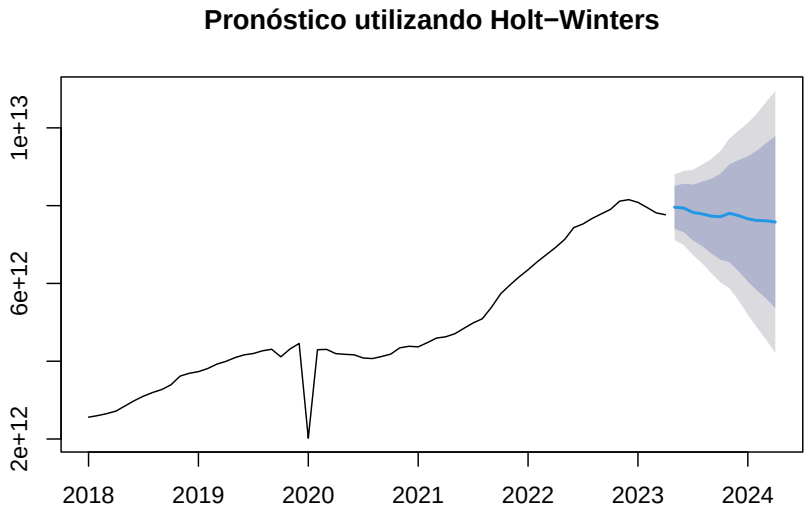


Al aplicar estas transformaciones, se espera que la serie de tiempo transformada tenga una tendencia y variabilidad reducidas en comparación con la serie original.

Metodología de Holt-Winters:

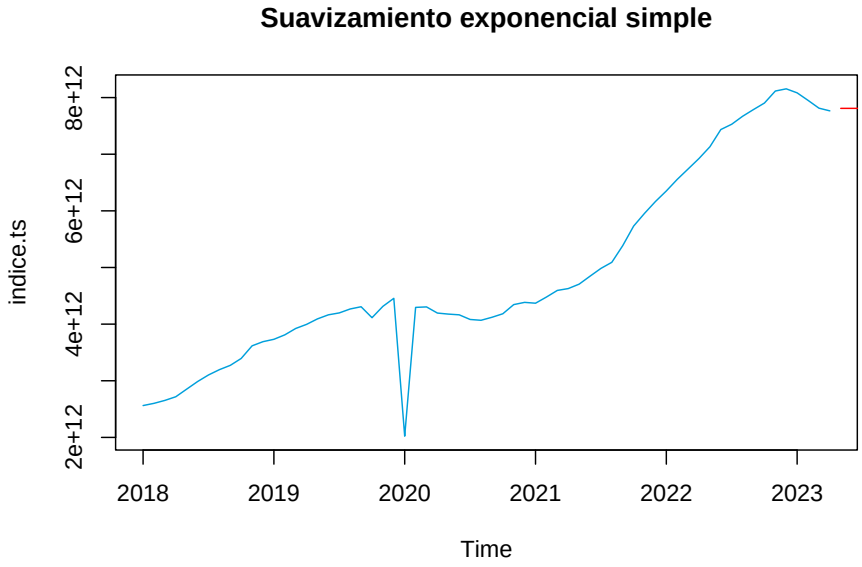
Para aplicar la metodología de Holt-Winters y suavizamiento exponencial a la serie de tiempo indice.ts, podemos utilizar las funciones correspondientes de R. La metodología de Holt-Winters es adecuada para modelar series de tiempo con componentes de tendencia y estacionalidad. El resultado del modelo Holt-Winters proporciona las componentes de tendencia (trend), estacionalidad (seasonal) y residuos (residuals). Estas componentes ayudan a comprender los patrones y la estructura de la serie de tiempo.

```
library(forecast)

# Aplicar la metodología de Holt-Winters
hw_model <- HoltWinters(indice.ts)

# Obtener el pronóstico utilizando el modelo Holt-Winters
hw_forecast <- forecast(hw_model, h = 12)  # Pronóstico para los próximos 12 periodos

# Graficar la serie de tiempo y el pronóstico
plot(hw_forecast, main = "Pronóstico utilizando Holt-Winters")
```



Además de la metodología de Holt-Winters, también puedes aplicar técnicas de suavizamiento para obtener una versión suavizada de la serie de tiempo. Una técnica común es el suavizamiento exponencial simple, que calcula un promedio ponderado de los valores anteriores para obtener la versión suavizada de la serie.Se Puede utilizar la función ses() de la librería forecast para aplicar esta técnica:

```
# Aplicar suavizamiento exponencial simple
ses_model <- ses(indice.ts)

# Obtener la serie suavizada
horizon <- 6  # Pronosticar 6 períodos hacia adelante
ses_smooth <- forecast(ses_model, h = horizon)$mean

# Graficar la serie de tiempo y la versión suavizada
plot(indice.ts, type = "l", col = "#00a0dc", main = "Suavizamiento exponencial simple")
lines(ses_smooth, col = "red")
```



El resultado del suavizamiento exponencial simple se muestra en el gráfico que se generó. Aquí hay una explicación de lo que puedes observar en el resultado:

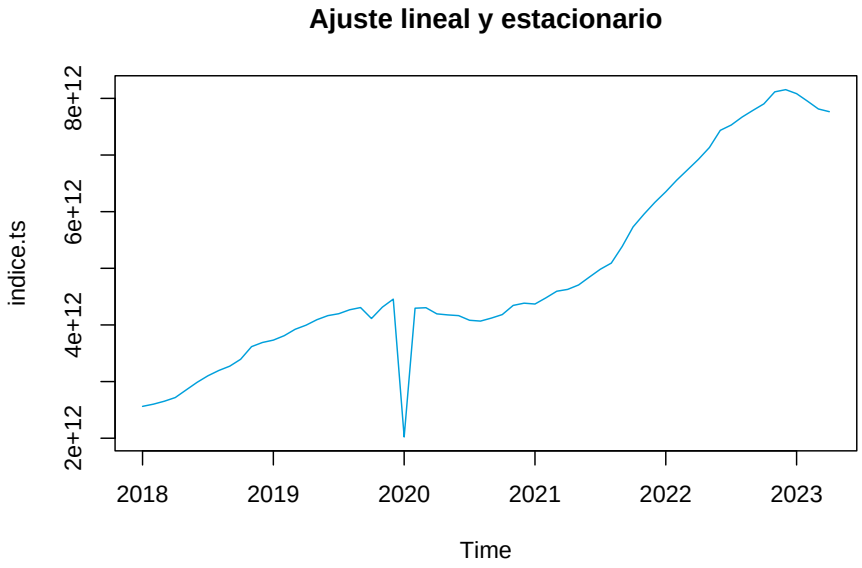
La línea azul representa la serie de tiempo original, es decir, los valores observados de la variable a lo largo del tiempo. La línea roja representa la versión suavizada de la serie de tiempo, obtenida mediante el suavizamiento exponencial simple. El suavizamiento exponencial simple utiliza un promedio ponderado de los valores anteriores para generar la versión suavizada. A medida que avanzas en el tiempo, la línea roja muestra una representación suavizada de la tendencia general de la serie. La versión suavizada ayuda a eliminar el ruido y las fluctuaciones más pequeñas presentes en la serie original. Esto permite visualizar mejor la tendencia subyacente en los datos. Observa cómo la línea roja se ajusta a la tendencia general de la serie original. En general, el suavizamiento exponencial simple puede ser útil para identificar patrones de tendencia en los datos y proporcionar una representación más clara de la evolución de la variable a lo largo del tiempo.

Modelos estacionarios en series de tiempo

Enfoque de regresión lineal clásico:

```
# Crear una variable de tiempo numérica
time <- 1:length(indice.ts)

# Ajustar un modelo lineal y estacionario
lm_model <- lm(indice.ts ~ time)
```

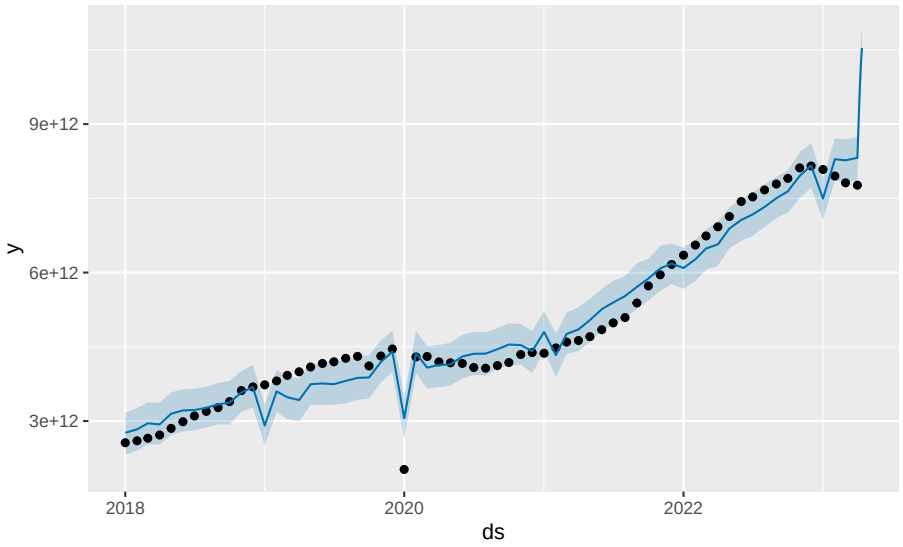
```r
# Obtener los coeficientes del modelo
coef_lm <- coef(lm_model)

# Graficar la serie de tiempo y el modelo ajustado
plot(indice.ts, type = "l", col = "#00a0dc", main = "Ajuste lineal y estacionario")
lines(fitted(lm_model), col = "red")
```



Algoritmo Facebook´s Prophet:

```r
# Instalar y cargar la librería prophet
#install.packages("prophet")
library(prophet)

# Crear un dataframe con la serie de tiempo
df <- data.frame(ds = as.Date(datos$Periodo),
                 y = datos$Saldo)

# Ajustar el modelo Prophet
prophet_model <- prophet(df)

# Realizar un pronóstico para los próximos 12 meses
future <- make_future_dataframe(prophet_model, periods = 12)
forecast <- predict(prophet_model, future)
```

```
# Graficar el pronóstico
plot(prophet_model, forecast)
```



Chapter 7

Modelos de Redes Neuronales Recurrentes

7.1 Elman

7.1.1 Carga de Datos

Para empezar el ejercicio, vamos a cargar nuevamente nuestros datos fuente. Esto con el fin de evitar cualquier transformacion anterior que pueda distorsionar la serie de tiempo original.

```
library(forecast)
library(timsac)
library(ggplot2)
library(changepoint)
library(readxl)
library(RSNNS)
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: TTR
```

```
datos <- read_excel("./fuente/BASE_Clientes.xlsx")
datos
```

```
## # A tibble: 643,570 x 8
```

```
##    Periodo Sub_Tipo N_Clientes DIAS_DE_MORA     Saldo Genero grupo_actividad_eco
##    <chr>   <chr>         <dbl>        <dbl>     <dbl> <chr>  <chr>
##  1 2018-01 CDC               4            0 15824105. Femen~ Dependiente privado
##  2 2018-01 CDC               1            0  6810373. Femen~ Dependiente privado
##  3 2018-01 CDC               6            6 28819502. Femen~ Dependiente privado
##  4 2018-01 CDC              12           63 81343674. Femen~ Dependiente privado
##  5 2018-01 CDC               1           21  7524344. Femen~ Dependiente privado
##  6 2018-01 CDC               4            0 12974213. Femen~ Dependiente privado
##  7 2018-01 CDC               3            1 21348609. Femen~ Dependiente privado
##  8 2018-01 CDC               2            0 11475858. Femen~ Dependiente privado
##  9 2018-01 CDC              10           38 60012355. Femen~ Dependiente privado
## 10 2018-01 CDC               1            0  9034715. Femen~ Dependiente privado
## # i 643,560 more rows
## # i 1 more variable: Cuidad_res <chr>
```

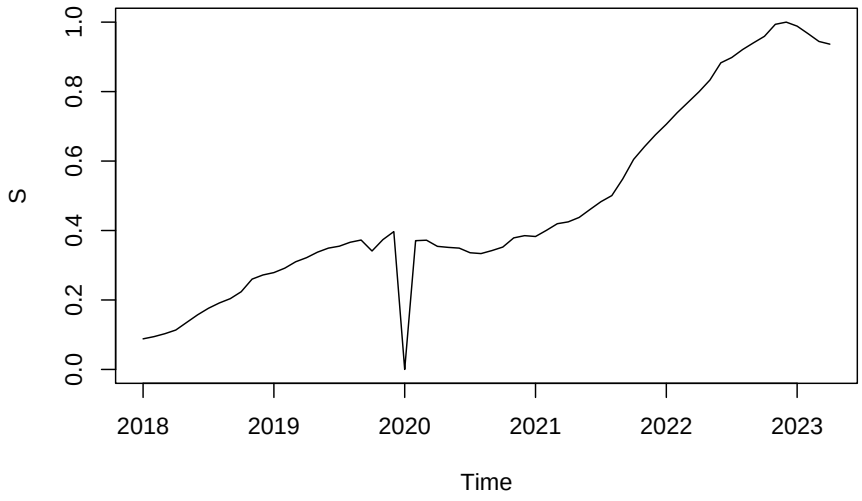
```r
# Cambio el tipo de dato de la columna temporal(Periodo)
datos$Periodo <- as.Date(paste0(datos$Periodo, "-01"))

# Consolido el df en funcion de la variable de interes (Saldo)
datos <- aggregate(Saldo ~ Periodo, data = datos, sum)

# Genero mi objeto ts para el analisis
indice.ts <- ts(datos$Saldo, start = c(2018,1), frequency = 12)
indice.ts
```

```
##               Jan          Feb          Mar          Apr          May
## 2018 2.562223e+12 2.601532e+12 2.653315e+12 2.717915e+12 2.852608e+12
## 2019 3.732137e+12 3.810740e+12 3.923380e+12 3.995810e+12 4.092819e+12
## 2020 2.022405e+12 4.295886e+12 4.304185e+12 4.195404e+12 4.177219e+12
## 2021 4.369300e+12 4.478002e+12 4.595047e+12 4.627874e+12 4.705783e+12
## 2022 6.351501e+12 6.555819e+12 6.739823e+12 6.925001e+12 7.132542e+12
## 2023 8.083445e+12 7.951094e+12 7.812947e+12 7.765132e+12
##               Jun          Jul          Aug          Sep          Oct
## 2018 2.986446e+12 3.102493e+12 3.196138e+12 3.272388e+12 3.394038e+12
## 2019 4.164386e+12 4.198177e+12 4.267921e+12 4.307340e+12 4.113506e+12
## 2020 4.164434e+12 4.082507e+12 4.067948e+12 4.120926e+12 4.182877e+12
## 2021 4.846473e+12 4.983882e+12 5.091594e+12 5.385784e+12 5.730839e+12
## 2022 7.435792e+12 7.529328e+12 7.670212e+12 7.789046e+12 7.903472e+12
## 2023
##               Nov          Dec
## 2018 3.617129e+12 3.690229e+12
## 2019 4.314034e+12 4.455499e+12
## 2020 4.344709e+12 4.384188e+12
## 2021 5.955740e+12 6.164705e+12
## 2022 8.115444e+12 8.154174e+12
## 2023
```

7.1.2 Creacion de la TS normalizada

Una vez hecho esto, convierto mis datos a una time series (Z). Como estaremos trabajando con Redes Neuronales Recurrentes, normalizaremos la serie para que sus datos fluctuen entre 0 y 1.

```
Z <- as.ts(indice.ts,F)
S <- (Z-min(Z))/(max(Z)-min(Z))
plot(S)
```



7.1.3 Division de la serie.

Ahora dividiremos el el numero de filas totales para trabajar con nuestros sets de entrenamiento y testing.

```
lineas_totales <-length(S)
t_train <- round(lineas_totales*0.75, digits=0)
l_train <- 0:(t_train-1)
t_test <- (t_train):lineas_totales
t_test
```

```
##  [1] 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
```

7.1.4 Creacion de nodos.

Ahora crearemos un df con los nodos que adelantaran un valor en el futuro

```
y <- as.zoo(S)
x1 <- Lag(y, k = 1)
x2 <- Lag(y, k = 2)
x3 <- Lag(y, k = 3)
x4 <- Lag(y, k = 4)
x5 <- Lag(y, k = 5)
x6 <- Lag(y, k = 6)
x7 <- Lag(y, k = 7)
x8 <- Lag(y, k = 8)
x9 <- Lag(y, k = 9)
x10 <- Lag(y, k = 10)
x11 <- Lag(y, k = 11)
x12 <- Lag(y, k = 12)
slogN <- cbind(y,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12)
# Elimenemos los valores desplazados
slogN <- slogN[-(1:12),]
```

Acto seguido, especificaremos los inputs y outputs de la red

```
inputs <- slogN[,2:13]
outputs <- slogN[,1]
```

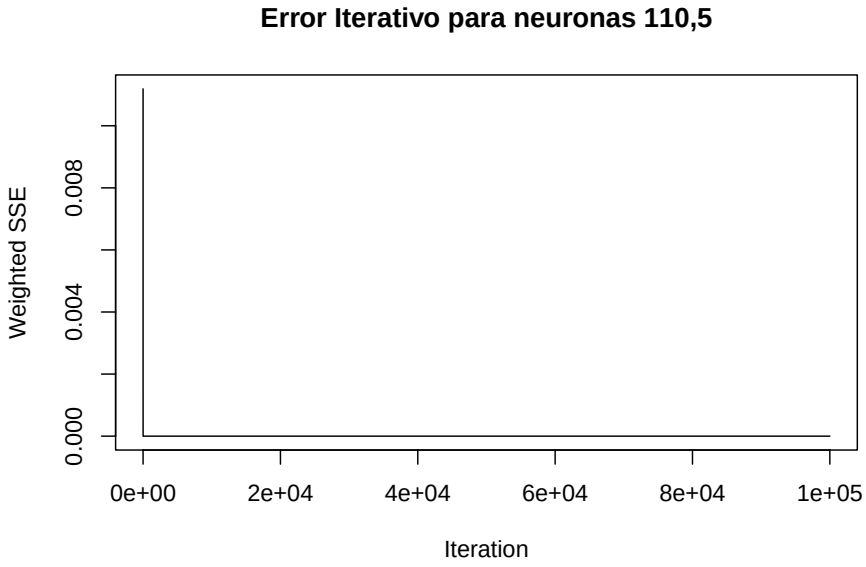
7.1.5 Entrenamiento del modelo Elman

Con la informaion en su lugar, procedamos a crear la red de Elman. Despues de varias pruebas, el numero de neuronas y profundidad queda en 110 y 5 respectivamente.

```
set.seed(42)
fit <- fit<-elman(inputs[t_train],outputs[t_train],size=c(110,5),learnFuncParams=c(0.1
```

Veamos como evoluciona el error

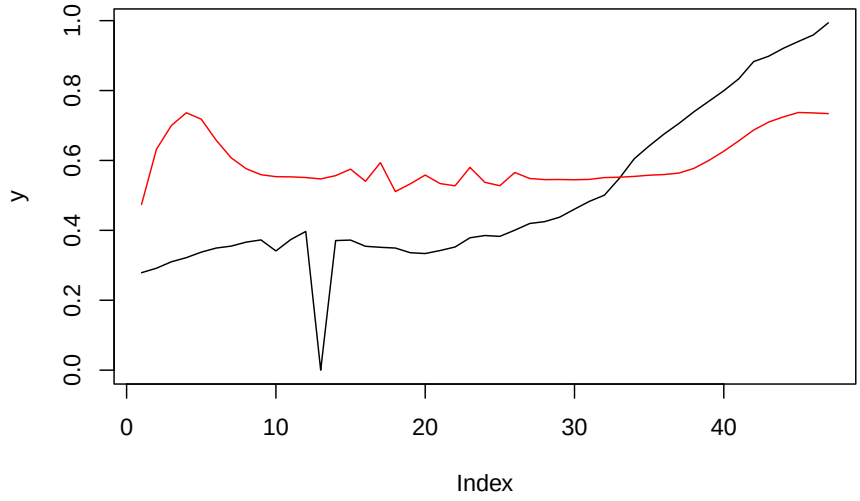
```
plotIterativeError(fit, main = "Error Iterativo para neuronas 110,5")
```



Como podemos observar, el error converge extremadamente rapido.

Ahora procederemos a hacer la la prediccion con el resto de terminos de la serie:

```
y <- as.vector(outputs[-t_test])
plot(y,type="l")
pred <- predict(fit, inputs[-t_test])
lines(pred,col = "red")
```



Prediccion

Procedemos a hacer las predicciones con nuestra red

```
predictions <- predict(fit,inputs[-t_train])
```

Desnormalizamos datos

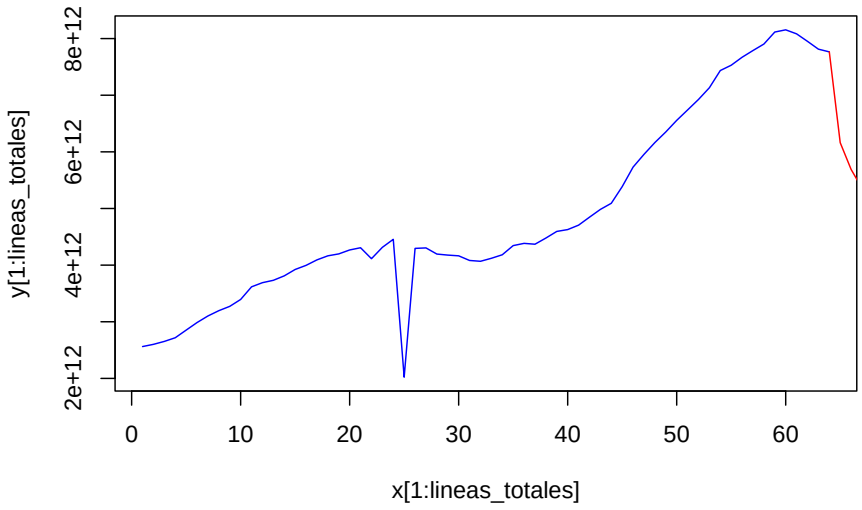
```
mod_elman <- predictions*(max(Z)-min(Z))+min(Z)
mod_elman
```

```
##                  [,1]
## Jan 2019 6.162340e+12
## Feb 2019 5.689504e+12
## Mar 2019 5.357780e+12
## Apr 2019 5.185718e+12
## May 2019 5.130582e+12
## Jun 2019 5.080404e+12
## Jul 2019 5.083780e+12
## Aug 2019 5.127912e+12
## Sep 2019 5.149160e+12
## Oct 2019 5.186750e+12
## Nov 2019 5.231140e+12
## Dec 2019 5.260487e+12
## Jan 2020 5.266360e+12
```

```
## Feb 2020 5.352791e+12
## Mar 2020 5.497074e+12
## Apr 2020 5.287112e+12
## May 2020 5.651816e+12
## Jun 2020 5.150608e+12
## Jul 2020 5.294518e+12
## Aug 2020 5.452413e+12
## Sep 2020 5.309498e+12
## Oct 2020 5.267031e+12
## Nov 2020 5.588612e+12
## Dec 2020 5.326231e+12
## Jan 2021 5.264540e+12
## Feb 2021 5.495523e+12
## Mar 2021 5.388697e+12
## Apr 2021 5.367434e+12
## May 2021 5.368399e+12
## Jun 2021 5.363104e+12
## Jul 2021 5.369137e+12
## Aug 2021 5.400116e+12
## Sep 2021 5.406626e+12
## Oct 2021 5.420363e+12
## Nov 2021 5.441833e+12
## Dec 2021 5.453365e+12
## Jan 2022 5.479368e+12
## Feb 2022 5.561483e+12
## Mar 2022 5.700627e+12
## Apr 2022 5.861865e+12
## May 2022 6.044024e+12
## Jun 2022 6.233664e+12
## Jul 2022 6.373705e+12
## Aug 2022 6.465632e+12
## Sep 2022 6.542706e+12
## Oct 2022 6.535043e+12
## Nov 2022 6.523112e+12
## Jan 2023 6.533344e+12
## Feb 2023 6.531746e+12
## Mar 2023 6.557522e+12
## Apr 2023 6.571427e+12
```

veamos los valores

```
x <- 1:(lineas_totales+length(mod_elman))
y <- c(as.vector(Z),mod_elman)
plot(x[1:lineas_totales], y[1:lineas_totales],col = "blue", type="l")
lines( x[(lineas_totales):length(x)], y[(lineas_totales):length(x)], col="red")
```



7.2 Jordan

7.2.1 Entrenamiento del modelo Jordan

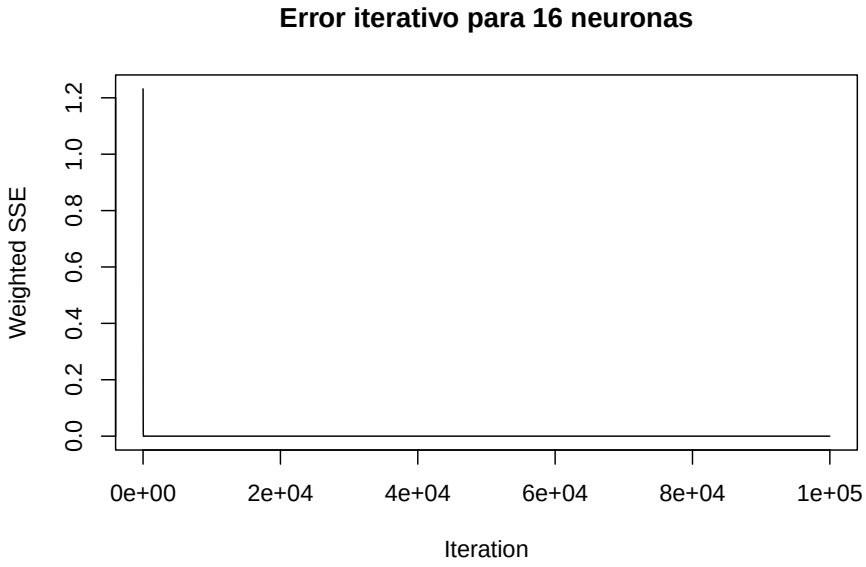
Como ya tenemos la tsnormalizada (S) y dividida entre training y testing, procedemos a entrenar el algoritmo con Jordan:

```
fit<-jordan(inputs[t_train],
    outputs[t_train],
    size=16,
    learnFuncParams=c(0.01),
    maxit=100000)
```

7.2.2 Error Iterativo

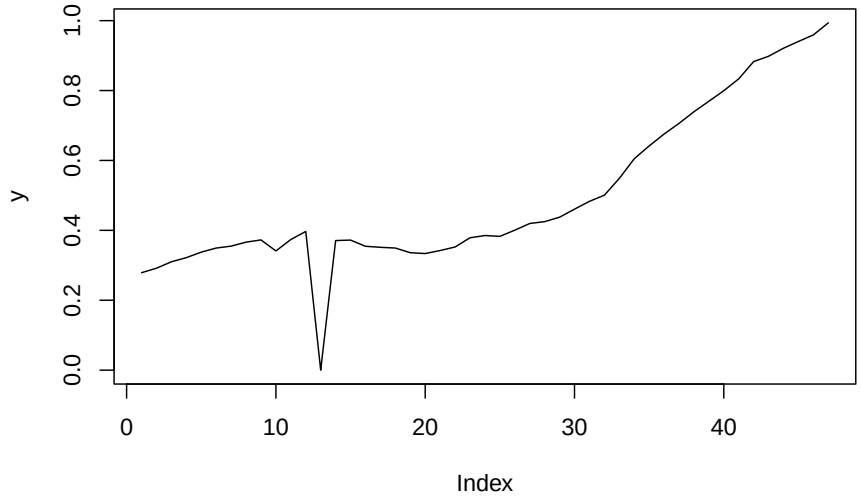
Ploteamos el error iterativo de 16 neuronas:

```
plotIterativeError(fit, main = "Error iterativo para 16 neuronas")
```



Veamos ahora como se comporta el error:

```
y <- as.vector(outputs[-t_test])
plot(y,type="l")
pred <- predict(fit, inputs[-t_test])
lines(pred,col = "red")
```



Procedemos con la prediccion

```
predictions <- predict(fit,inputs[-t_train])
mod_jordan <- predictions*(max(Z)-min(Z))+min(Z)
mod_jordan
```

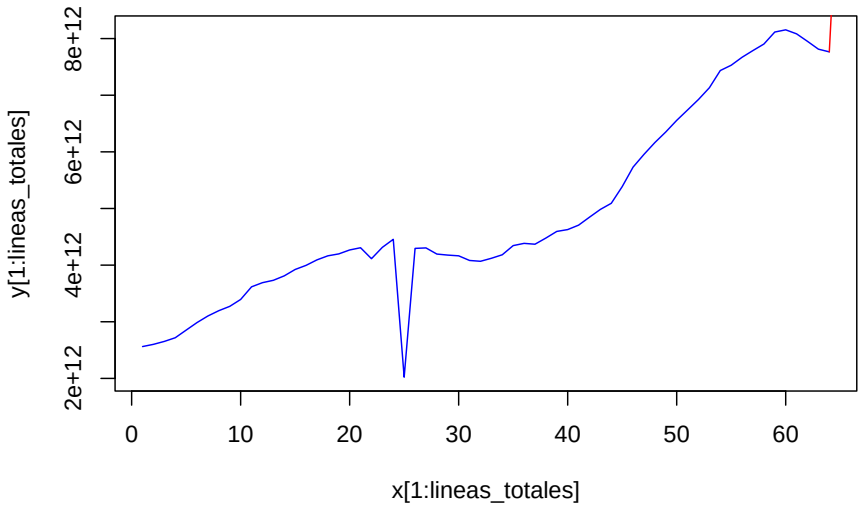
```
##                  [,1]
## Jan 2019 1.148488e+13
## Feb 2019 1.141995e+13
## Mar 2019 1.139538e+13
## Apr 2019 1.134228e+13
## May 2019 1.129364e+13
## Jun 2019 1.128617e+13
## Jul 2019 1.130867e+13
## Aug 2019 1.133650e+13
## Sep 2019 1.133135e+13
## Oct 2019 1.129645e+13
## Nov 2019 1.137222e+13
## Dec 2019 1.128734e+13
## Jan 2020 1.126138e+13
## Feb 2020 1.186075e+13
## Mar 2020 1.069085e+13
## Apr 2020 1.085286e+13
## May 2020 1.086688e+13
## Jun 2020 1.095609e+13
```

```
## Jul 2020 1.168398e+13
## Aug 2020 1.113439e+13
## Sep 2020 1.118931e+13
## Oct 2020 1.121318e+13
## Nov 2020 1.079012e+13
## Dec 2020 1.189949e+13
## Jan 2021 1.087328e+13
## Feb 2021 1.122287e+13
## Mar 2021 1.127449e+13
## Apr 2021 1.128306e+13
## May 2021 1.129050e+13
## Jun 2021 1.134053e+13
## Jul 2021 1.136667e+13
## Aug 2021 1.135414e+13
## Sep 2021 1.138047e+13
## Oct 2021 1.134630e+13
## Nov 2021 1.137113e+13
## Dec 2021 1.146756e+13
## Jan 2022 1.154174e+13
## Feb 2022 1.163130e+13
## Mar 2022 1.168323e+13
## Apr 2022 1.170141e+13
## May 2022 1.173339e+13
## Jun 2022 1.176145e+13
## Jul 2022 1.179491e+13
## Aug 2022 1.183078e+13
## Sep 2022 1.185394e+13
## Oct 2022 1.190688e+13
## Nov 2022 1.191501e+13
## Jan 2023 1.193379e+13
## Feb 2023 1.197258e+13
## Mar 2023 1.198780e+13
## Apr 2023 1.195651e+13
```

7.2.3 Resultados de la prediccion

Ahora veamoslo en la grafica

```
x <- 1:(lineas_totales+length(mod_jordan))
y <- c(as.vector(Z),mod_jordan)
plot(x[1:lineas_totales], y[1:lineas_totales],col = "blue", type="l")
lines( x[(lineas_totales):length(x)], y[(lineas_totales):length(x)], col="red")
```


y[1:lineas_totales]
2e+12
4e+12
6e+12
8e+12
0
10
20
30
40
50
60
x[1:lineas_totales]