Máster Universitario en Big Data y Ciencia de Datos

Actividad 2 - Estadística Avanzada

Alumno: Sellés Pérez, Alejandro

Edición Octubre 2024 a 12/03/2024

Índice

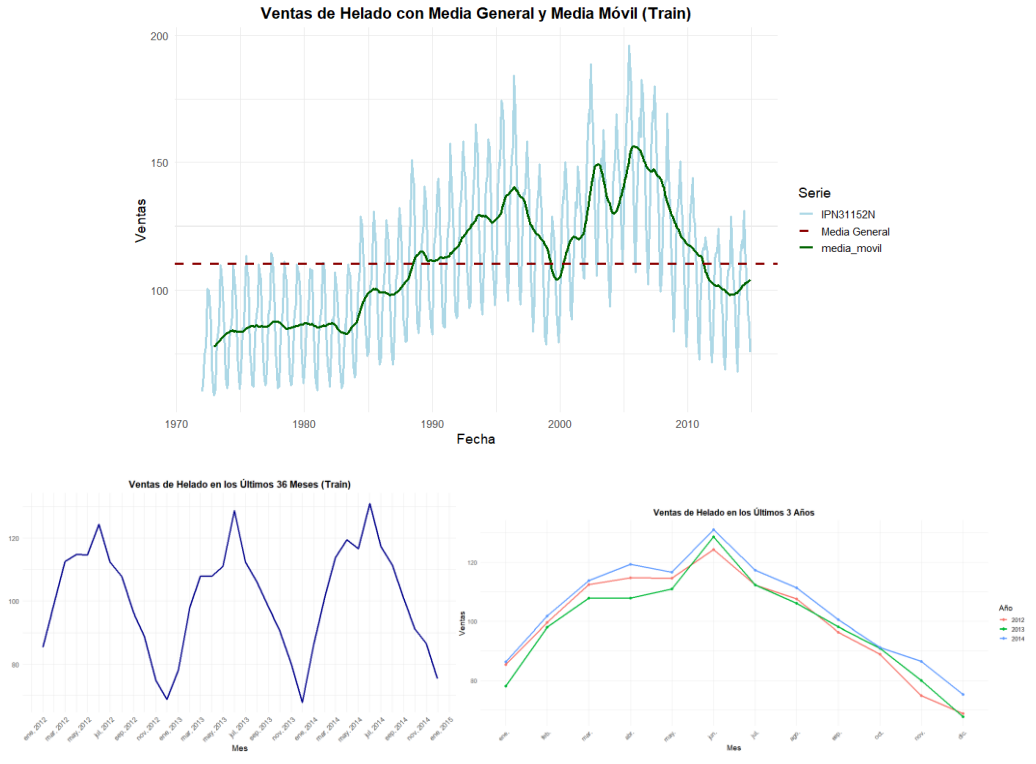
1. Introducción

A lo largo de esta actividad, nos vamos a poner en la piel de “Gelateria El Italiano”, una gran franquicia de heladerías española. El objetivo del proyecto será elaborar un modelo de series temporales capaz de predecir el número de helados que venderán en un futuro, para así poder realizar un buen control de inventario y anticiparse a posibles picos (altos o bajos) de demanda.

1.1. Descripción del Dataset

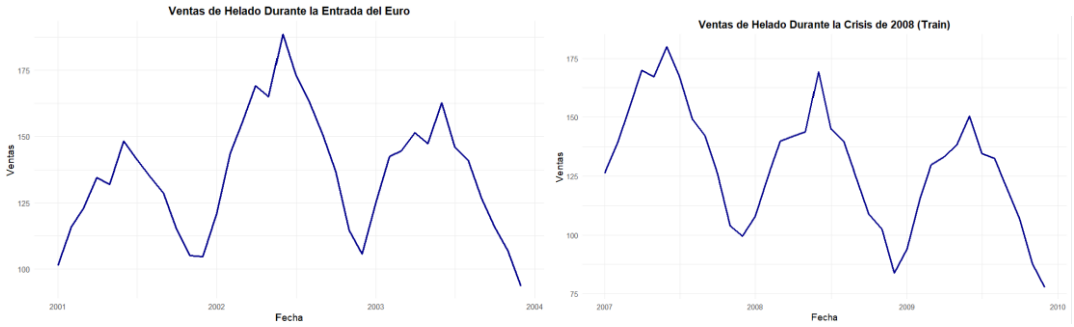
El dataset lo hemos obtenido de Kaggle (https://www.kaggle.com/datasets/abdocan/monthly-ice-cream-sales-data-1972-2020). Vamos a suponer que contamos con datos recopilados de una tienda ubicada en Alicante. Tenemos entonces datos diarios que van desde 1972, momento en el que se fundó, hasta el 2020.

En primer lugar, para realizar nuestro análisis, vamos a realizar una división train-test de nuestros datos. Tomaremos como train los datos desde 1972 hasta el 2014, dejando así una ventana de 5 años (2015-2020) para el test. La idea de realizar esta división será entrenar y obtener los coeficientes de los modelos con la parte del train, mientras que con los datos de test analizaremos cómo predicen los modelos y cuál de ellos es el más fiable. Vamos ahora a analizar un poco cómo se comportan nuestros datos de train. Para ello, comenzamos pintando la serie temporal completa, haciendo zoom en los últimos 3 años:

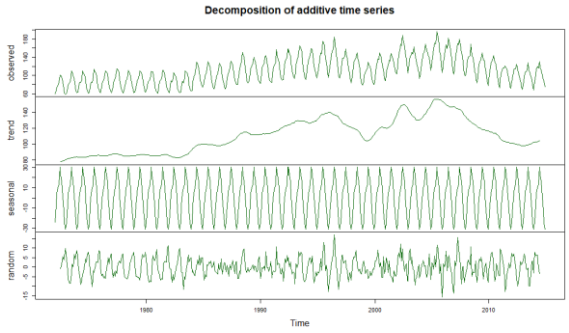


Podemos ver como tenemos unas ventas anuales bastante estables, con picos por temporadas, cosa que es normal ya que es lógico que en los meses de buen clima (abril – agosto) vendamos más helados que en invierno. Podemos afirmar esta hipótesis analizando los datos de los últimos 3 años, donde podemos observar también, que nuestro mes estrella es Junio. Además, podemos observar que crecimos bastante en las décadas de los 1990 y 2000, pero al inicio de los 2000 y en los últimos años ha habido una bajada en las ventas. Si nos fijamos, podemos deducir esto coincide con 2 momentos clave en la economía española: la entrada del euro y la crisis de 2008. En las siguientes gráficas podemos observar cómo la entrada del euro

no parece muy influyente, mientras que en la crisis sí hay una tendencia decreciente en las ventas:



Ahora, para terminar con nuestro análisis, obtenemos las descomposición aditiva, donde podemos ver como en los primeros años la tendencia era claramente a vender más, y tras unos periodos de subidas y bajadas, en los últimos años de entrenamiento tenemos una fuerte bajada. Por lo tanto, a simple vista podríamos decir que la serie no es estacionaria.



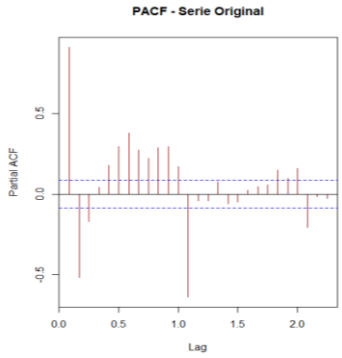
2. Modelos de Series Temporales

A lo largo de esta sección vamos a elaborar modelos de series temporales con el objetivo de predecir el número de ventas de nuestra tienda de helados. Como hemos visto en el apartado anterior, entrenaremos nuestros modelos con la parte del train, y con la del test analizaremos nuestras predicciones y compararemos los modelos obtenidos. En esta segunda parte, para realizar la evaluación de los modelos, obtendremos 2 predicciones diferentes:

- Predicción estática: Se obtiene utilizando solamente los datos que tenemos en nuestro conjunto de datos de entrenamiento.
- Predicción progresiva: Es la realmente interesante. Se obtiene utilizando además de los datos de nuestro conjunto de datos de entrenamiento, las predicciones que vamos obteniendo en puntos anteriores.

2.1. Modelo AR

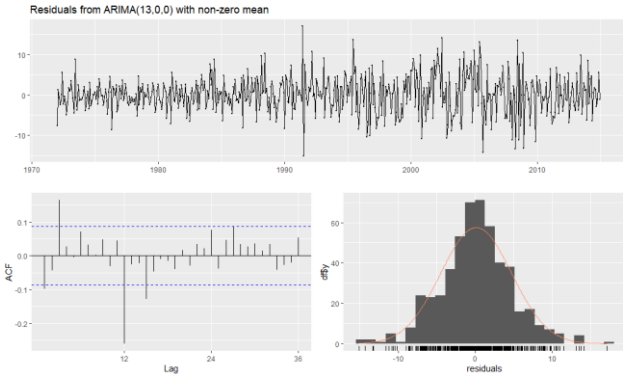
Comenzamos la búsqueda del mejor modelo predictivo con un modelo *AR(p)*. Para escoger el valor de *p* adecuado, vamos a pintar las autocorrelaciones parciales (PACF) de nuestra serie temporal de entrenamiento:



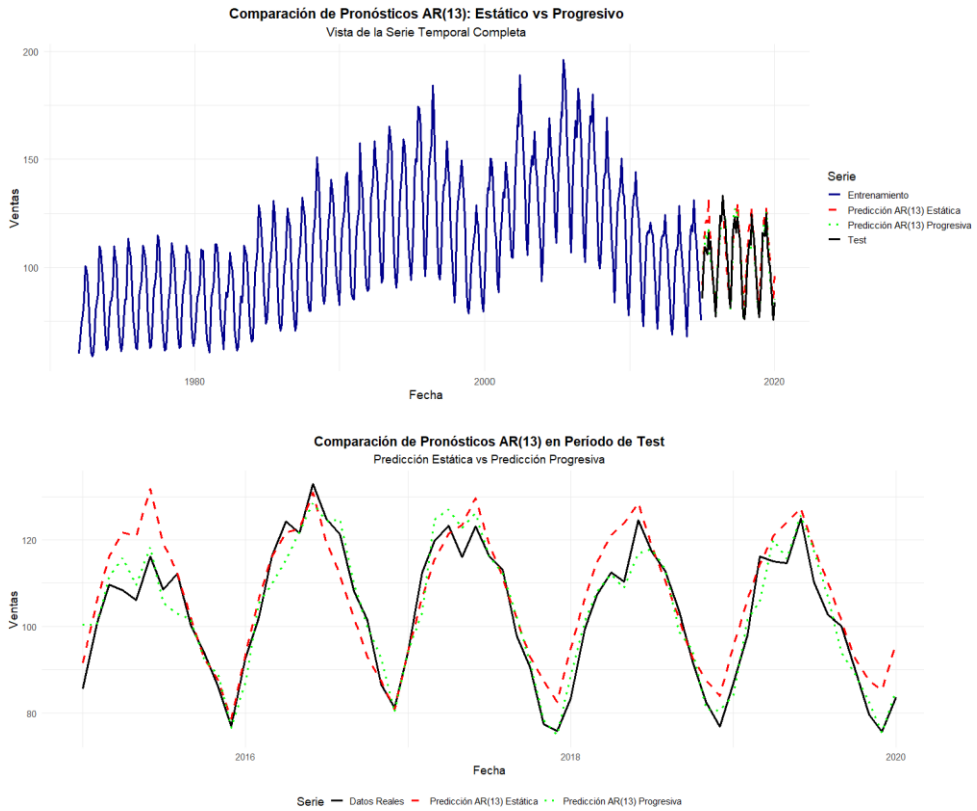
Podemos ver cómo los valores en valor absoluto de PACF parece que comienzan a descender a partir del lag = 3. Sin embargo, en lag = 13 tenemos un valor muy alto también, y a partir de ahí sí que podríamos decir que disminuye mucho. Por ello, vamos a escoger *p = 13*. Aplicamos el modelo y obtenemos los resultados:

```
Coefficients:
         ar1     ar2      ar3      ar4     ar5      ar6     ar7     ar8      ar9     ar10    ar11    ar12     ar13      mean
      0.8690  0.0556  -0.0227  -0.0444  0.0425  -0.0559  0.0033  0.0641  -0.0473  0.0017  0.1140  0.7364  -0.7327  106.2929
s.e.  0.0294  0.0354   0.0352   0.0351  0.0350   0.0346  0.0348  0.0352   0.0353  0.0353  0.0352  0.0352   0.0293   10.8942

sigma^2 = 21.87:  log likelihood = -1530.23
AIC=3090.46   AICc=3091.42   BIC=3154.15

Training set error measures:
                     ME     RMSE      MAE          MPE     MAPE      MASE         ACF1
Training set 0.1296911 4.612752 3.480145 -0.04650673 3.160551 0.4808862 -0.09744025
```

Vemos que tenemos un RMSE de 4.613 y un AIC de 3091.42 en nuestro conjunto de entrenamiento. Ahora, realizamos un análisis de los residuos obtenidos. Podemos ver que nuestros errores siguen una distribución bastante normal y se encuentran en el intervalo [-10,10]. Además, teniendo en cuenta que nuestros datos de ventas están en torno a unas 120 ventas diarias, podríamos decir que el modelo predice bastante bien.



Finalmente, evaluamos nuestro modelo en el conjunto de datos de test para analizar las predicciones que está haciendo. Obtenemos:



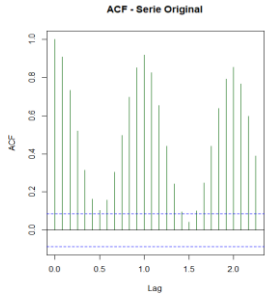
Podemos ver que el modelo se ajusta bastante bien a los periodos de subidas y bajadas en nuestro conjunto de test, y podemos ver que la principal diferencia entre la predicción estática y la progresiva es ese primer año 2015, donde la predicción progresiva es capaz de captar que la subida será más leve que otros años.

Finalmente, obtenemos los errores del modelo en la parte de entrenamiento, que utilizaremos posteriormente para comparar los modelos:

Métrica	Valor
RMSE estático	6.7299
RMSE progresivo	4.5432
MAE estático	5.3923
MAE progresivo	3.3641

2.2. Modelo MA

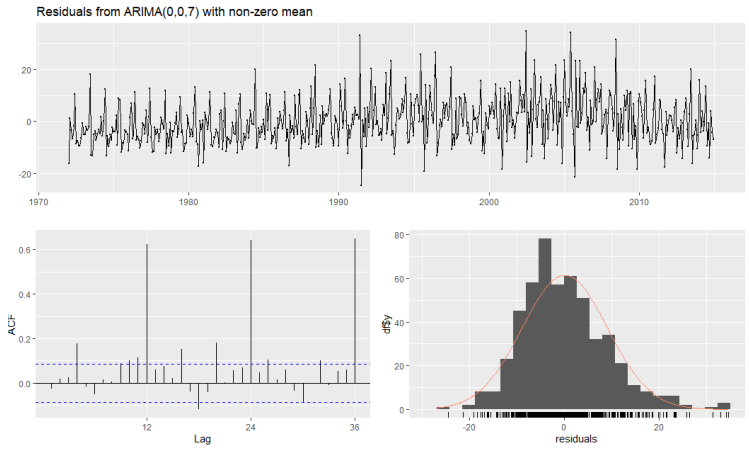
Vamos a seguir en la búsqueda del mejor modelo. En esta ocasión, vamos a buscar el mejor modelo *MA(q)*. Para obtener este valor de *q*, vamos a visualizar la autocorrelación (ACF) del conjunto de entrenamiento:



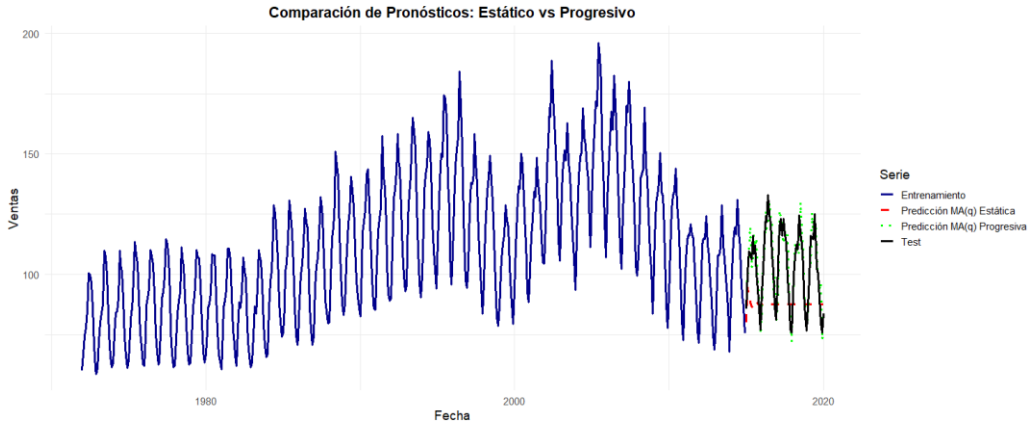
Podemos ver cómo no parece que baje en ningún momento mucho. De hecho, parece que hay correlaciones periódicas. Esto es probablemente porque los comportamientos se repiten bastante según el día de la semana en el que estemos. Observando la gráfica, vamos a tomar *q = 7,* ya que es el momento en el que tiene una primera bajada, para obtener nuestro modelo. Obtenemos:

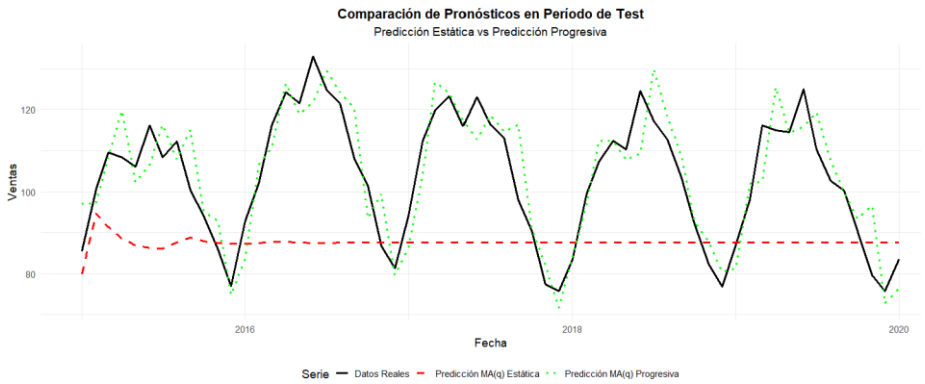
```
Coefficients:
         ma1     ma2     ma3     ma4     ma5     ma6      ma7       mean
      1.3570  1.5832  1.6285  1.0904  0.6322  0.1219  -0.2397  110.0242
s.e.  0.0415  0.0693  0.0886  0.0988  0.0925  0.0758   0.0422    2.8335

sigma^2 = 82.48:  log likelihood = -1869.36
AIC=3756.73   AICc=3757.08   BIC=3794.94

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE     MASE         ACF1
Training set 0.05253447 9.011091 7.007218 -0.7731732 6.38907 0.968257 -0.02428931
```

Esta vez tenemos un RMSE bastante peor al anterior, de 9.011, y un AIC más alto, de 3756.73. Además, analizando los residuos de la serie de entrenamiento, podemos ver cómo siguen una distribución bastante normal, y cómo obtenemos errores dentro de [-20,20]. Podemos ver claramente que tenemos algunos puntos con errores bastante mayores a los del modelo anterior.



Ahora, vamos a analizar las predicciones del modelo para la parte del test:





Por un lado, vemos que la predicción estática a partir de cierto punto se hace una recta, esto es porque evalúa con los mismos datos constantemente. Y, por otro lado, en la predicción progresiva podemos observar que nuestro modelo está captando bastante bien los picos de subidas y bajadas de nuestra serie, de manera bastante similar al modelo anterior.

Finalmente, mostramos las métricas de error para esta parte de train. Podemos ver que claramente tanto la predicción estática como la progresiva son peores que las del modelo anterior:

Métrica	valor
RMSE estático	16.5610
RMSE progresivo	7.6079
MAE estático	13.3774
MAE progresivo	6.0749

2.3. Modelo ARIMA

En esta sección vamos a buscar el mejor modelo ARIMA para predecir nuestra serie. En primer lugar, vamos a buscar el número de diferenciaciones (*d*) necesarias hasta llegar a una serie estacionaria. Para ello, realizaremos el test de Dickey-Fuller aumentado (ADF), el cual toma:

- Hipótesis nula (H_0): La serie temporal tiene una raíz unitaria y, por lo tanto, es no estacionaria (es decir, tiene una tendencia estocástica).
- Hipótesis alternativa (H_1): La serie temporal no tiene una raíz unitaria y, por lo tanto, es estacionaria.

Por tanto, tomando un nivel de significancia de 0.05, asumiremos que si el p-valor obtenido es menor que 0.05 la serie es estacionaria.

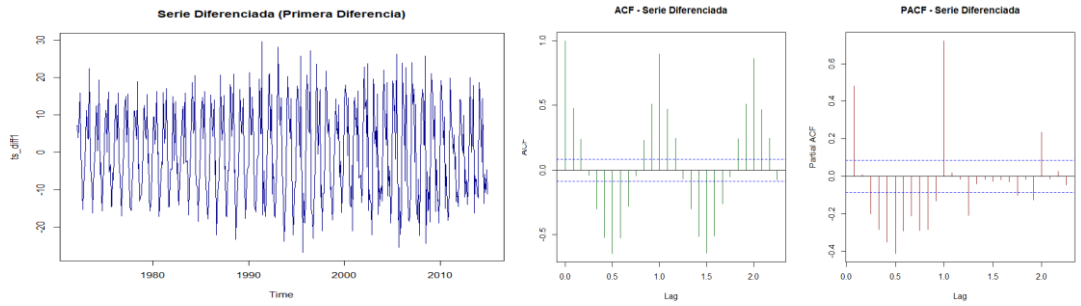
Comenzamos aplicando el test ADF a nuestra serie temporal de entrenamiento y vemos que el p-valor es de 0.6884, por lo que, como ya habíamos predicho en el primer apartado, no podemos afirmar que es estacionaria.

```
        Augmented Dickey-Fuller Test

data:  ts_train
Dickey-Fuller = -1.7399, Lag order = 8, p-value = 0.6884
alternative hypothesis: stationary
```

Ahora, realizamos una diferenciación y volvemos a probar la prueba ADF. En esta ocasión, obtenemos un p-valor menor que 0.05, por lo que asumiremos que la serie es estacionaria.

```
        Augmented Dickey-Fuller Test

data:  ts_diff1
Dickey-Fuller = -20.532, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary
```

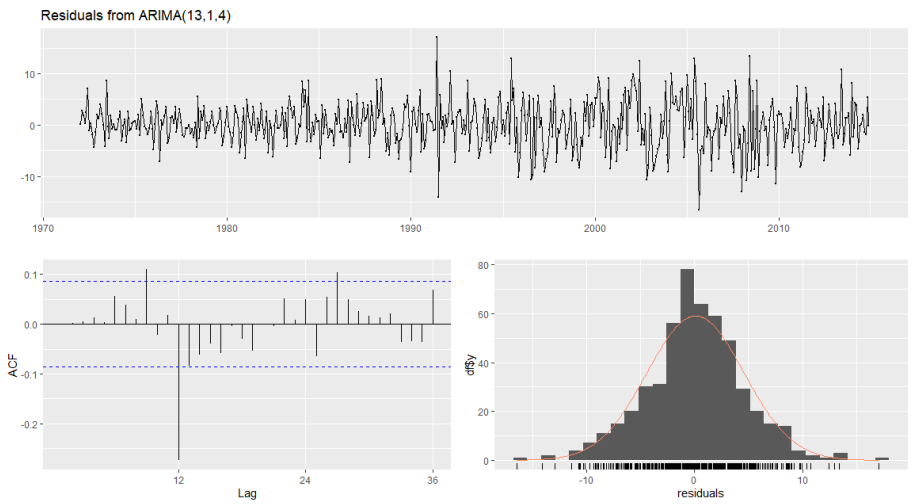
Por tanto, para nuestro modelo ARIMA vamos a tomar d = 1. Pintamos entonces la serie con una diferenciación y sus gráficas de ACF y PACF:



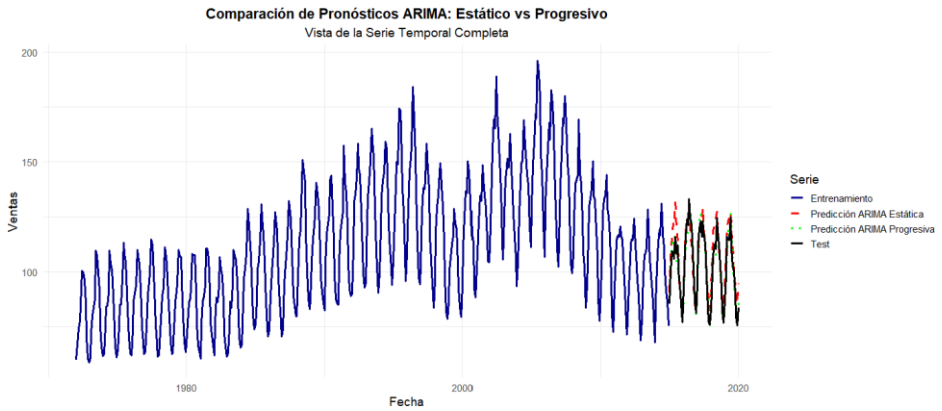
Analizando los momentos en los que descienden en valor absoluto los valores de ACF y PACF, vamos a tomar *p = 13* y *q = 4.* Por tanto, tomamos el modelo *ARIMA(13,1,4).* Obtenemos:

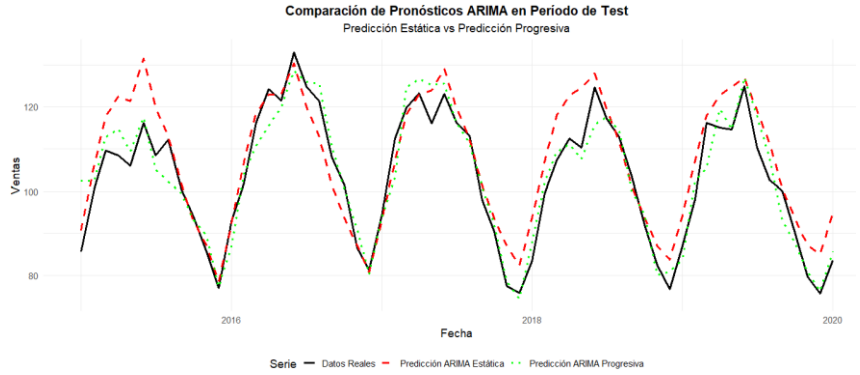
```
Coefficients:
          ar1      ar2      ar3      ar4      ar5      ar6      ar7      ar8      ar9     ar10     ar11     ar12     ar13     ma1
      -0.275  -0.0605  -0.1791  -0.1515  -0.1134  -0.1617  -0.1615  -0.0988  -0.1524  -0.1537  -0.029   0.7161   0.1318  0.0496
s.e.     NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN   0.0372      NaN     NaN
          ma2      ma3      ma4
      -0.0642   0.2458   0.0253
s.e.     NaN   0.0557      NaN

sigma^2 = 20.94:  log likelihood = -1513.37
AIC=3062.74   AICc=3064.12   BIC=3139.14

Training set error measures:
                     ME     RMSE      MAE         MPE     MAPE     MASE          ACF1
Training set 0.09212373 4.495877 3.401189 0.003599412 3.059933 0.469976 -0.0001807667
```

Podemos ver que tenemos un RMSE de 4.49, bastante inferior al obtenido con el modelo anterior, y muy similar al del primero modelo. Mientras que, el AIC es muy similar al obtenido en el primer modelo *AR(13)*. Tomando las gráficas de los residuos para la parte de entrenamiento, podemos ver que obtenemos resultados bastante mejores a los del resto de modelos. A simple vista, vemos que los puntos están más cercanos al 0 y la distribución es aún más normal, con menos picos.



Ahora, graficamos las predicciones de nuestros datos para la parte del test y obtenemos:





Podemos observar que los resultados son muy buenos. Nuestro modelo es capaz de captar bastante bien las subidas y bajadas. Sí es cierto que, del mismo modo que en el resto, la predicción estática no obtiene buenos resultados en el año 2015.

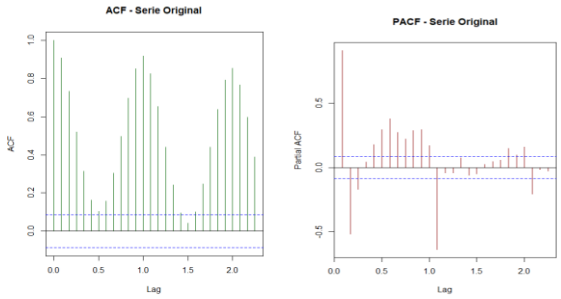
Finalmente, mostramos las métricas relevantes en la parte de test. Vemos que los resultados de nuestras predicciones son muy similares a los obtenidos con el primer modelo. Esto nos podría hacer pensar que quizás el otro modelo es mejor, ya que tiene una complejidad menor. Sin embargo, no vamos a sacar ninguna conclusión todavía. Más adelante, en la sección 3, nos centraremos en realizar estas comparaciones entre modelos.

Métrica	Valor
RMSE estático	6.8764
RMSE progresivo	4.7791
MAE estático	5.4843
MAE progresivo	3.6038

2.4. Modelo SARIMA

Para terminar, vamos a partir del modelo ARIMA obtenido, y vamos a añadirle las componentes estacionales.

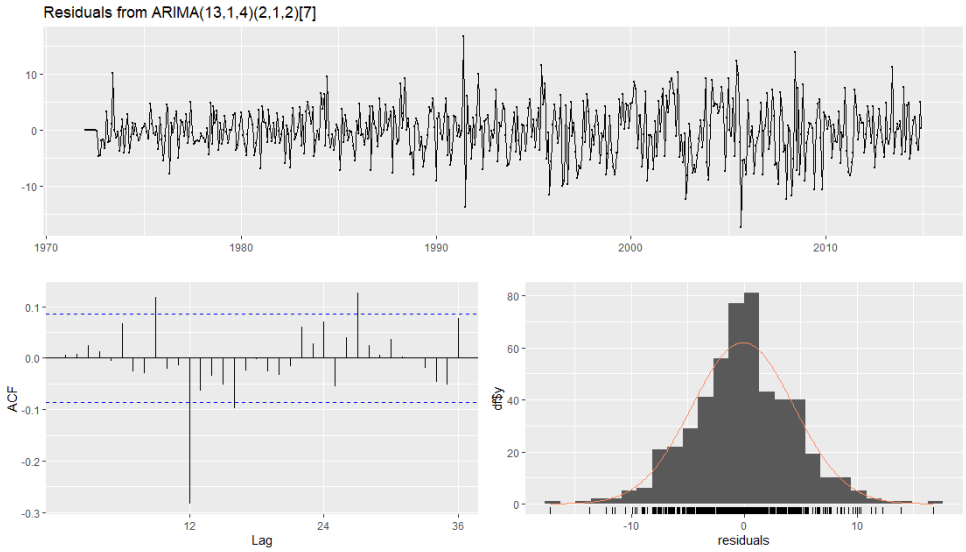
Para escoger los parámetros, me he fijado en las gráficas de ACF y PACF aneriores. Debido a que tenemos los datos por día, y analizando la gráfica del ACF, he decidido tomar un periodo de s = 7 días, con la idea de tener en cuenta en el modelo las ventas del mismo día de la semana (lunes, martes…) pero de semanas anteriores. Por otro lado, he tomado el valor de $P = 2$, ya que parece que después de 14 lags (2x7) ya no hay mucha relación en PACF. Además, he escogido el valor de $Q = 2$, ya que parece que sigue habiendo relaciones en ACF, pero no quería hacer un modelo excesivamente complejo. Finalmente, he escogido $D = 1$, ya que parece haber tendencia estacional pero la serie inicial no es estacionaria.



Por lo tanto, vamos a trabajar con el modelo *SARIMA((13,1,4),(2,1,2)[7])*. Obtenemos:

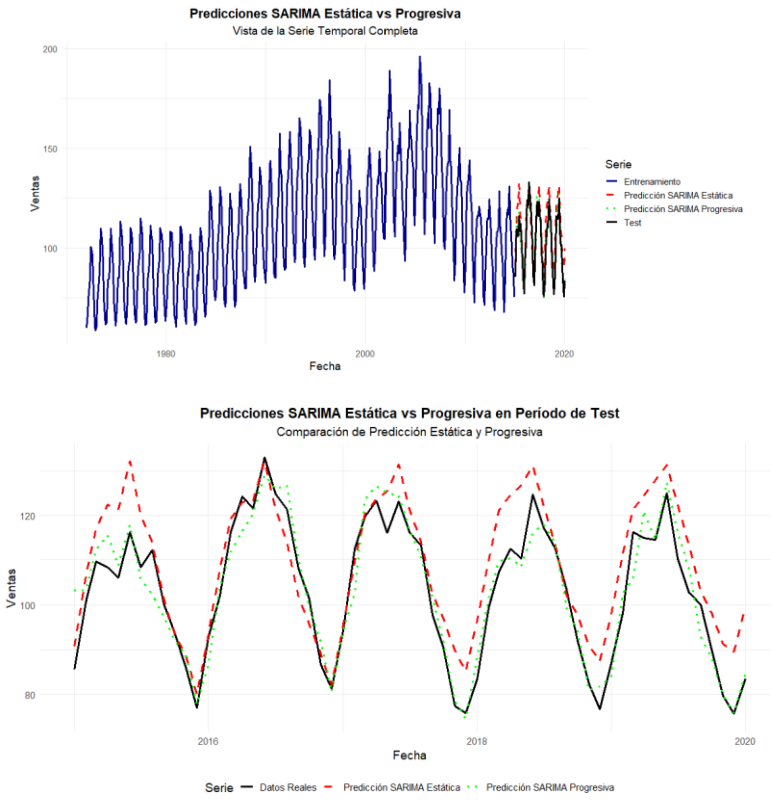
```
Coefficients:
          ar1     ar2      ar3     ar4      ar5     ar6      ar7      ar8     ar9     ar10    ar11    ar12    ar13    ma1      ma2     ma3     ma4     sar1     sar2    sma1     sma2
      -0.1546  0.0105  -0.1423  -0.0954  -0.0476  -0.1399  -0.1292  -0.0060  -0.1042  -0.1336  0.0404  0.7639  0.0380  -0.086  -0.1512  0.2234  -0.018  -0.1996  -0.0666  -0.7175  -0.2762
s.e.  0.0026     NaN   0.0011   0.0024      NaN      NaN      NaN   0.0023   0.0018      NaN  0.0028  0.0006  0.0012     NaN      NaN     NaN     NaN   0.0022   0.0007   0.0055   0.0059

sigma^2 = 21.43:  log likelihood = -1511.04
AIC=3066.08   AICc=3068.16   BIC=3159.15

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
Training set -0.1152125 4.497391 3.425207 -0.229909 3.107321 0.4732949 0.006263279
```

Podemos ver que el RMSE obtenido es de 4.497 y el AIC de 3066.08, resultados prácticamente iguales a los del *ARIMA* del apartado anterior. Veamos ahora cómo se comportan los residuos en esta parte de entrenamiento:



Podemos observar un comportamiento algo mejor al del modelo anterior. Fijándonos en el histograma, podemos ver que tenemos un pico mayor de valores muy cercanos al 0 por la izquierda.

Ahora, graficamos las predicciones de nuestros datos para la parte del test y obtenemos:



Podemos observar que los resultados son similares a los del modelo anterior, ajustándose muy bien a los resultados de test reales que tenemos.

Finalmente, mostramos las métricas más relevantes sobre esta predicción. Podemos decir que son resultados algo peores a los obtenidos anteriormente.

Table: Modelo: SARIMA

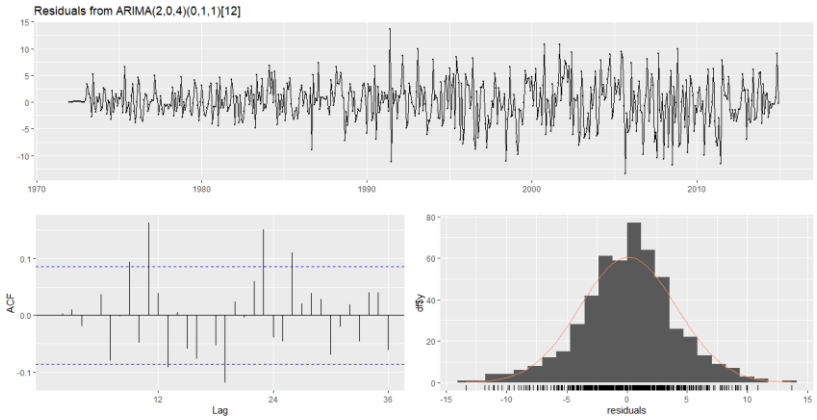
Métrica	Valor
RMSE estático	8.4463
RMSE progresivo	4.8141
MAE estático	6.9174
MAE progresivo	3.5981

2.5. Auto-Arima

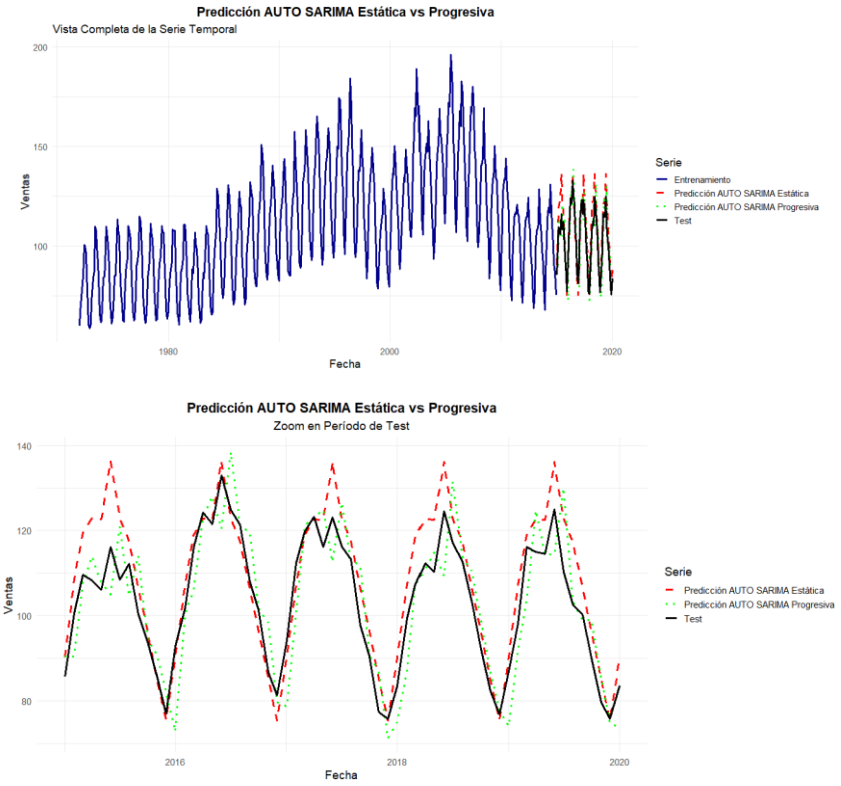
Para terminar con la búsqueda de modelos, vamos a utilizar la función de auto.arima en R, la cual busca automáticamente los mejores parámetros para el modelo basándose en el criterio AIC.

Ejecutamos la función en nuestro conjunto de entrenamiento y obtenemos:

```
Series: ts_train
ARIMA(2,0,4)(0,1,1)[12]

Coefficients:
         ar1      ar2      ma1     ma2     ma3      ma4     sma1
      1.4146  -0.4341  -0.6243  0.0274  0.2178  -0.1704  -0.6964
s.e.  0.3933   0.3763   0.3902  0.0796  0.0570   0.0693   0.0375

sigma^2 = 16.53:  log likelihood = -1423.11
AIC=2862.22   AICc=2862.51   BIC=2896

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set 0.1008272 3.989832 3.034191 0.0894185 2.726701 0.4192644 9.517647e-05
```



Hemos llegado a un modelo *SARIMA((2,0,4),(0,1,1)[12]).* Si nos fijamos, hemos obtenido un modelo bastante más simple que los que habíamos probado anteriormente, y que además tiene un RMSE menor y una distribución de los residuos bastante mejor para el conjunto de entrenamiento. Veamos ahora cómo predice con nuestros datos de test:



Podemos ver cómo se ajusta muy bien a los datos de test. Sin embargo, a simple vista no podemos concluir si funciona mejor que los modelos anteriores. Por ello, en la siguiente sección vamos a elaborar una tabla de comparación de las métricas de cada uno de los modelos obtenidos.

3. Conclusiones

Para terminar nuestro análisis, vamos a elaborar una tabla de comparación de todos los modelos. Obtenemos:

Modelo	AIC	RMSE_Train	RMSE_Test_Static	RMSE_Test_Rolling	MAE_Test_Static	MAE_Test_Rolling
AR	3090.457	4.6766	6.7299	4.5432	5.3923	3.3641
MA	3756.727	9.0818	16.5610	7.6079	13.3774	6.0749
ARIMA	3062.744	4.5764	6.8764	4.7791	5.4843	3.6038
SARIMA	3066.077	4.6294	8.4463	4.8141	6.9174	3.5981
AUTO SARIMA	2862.219	4.0654	7.6736	8.9218	6.2395	7.1108

Podemos ver que, para la parte del entrenamiento, el mejor modelo obtenido es el Auto SARIMA. Sin embargo, si nos fijamos en los errores para la parte del test de las predicciones progresivas, podemos ver que el mejor modelo es el primer AR obtenido, y podemos observar que el Auto SARIMA se queda bastante atrás en comparación con el resto.

Por ello, a la hora de escoger el modelo, podríamos basarnos en la complejidad del modelo, donde quizá lo mejor sería simplemente quedarnos con el *AR(13)*; o, por otro lado, optar por un modelo más complejo pero bastante fiable también, como el ARIMA que hemos construido.

4. Siguientes pasos

Para mejorar estos resultados, en futuros trabajos sería recomendable explorar otros modelos de series temporales como Prophet de Meta o Redes Neuronales Recurrentes (LSTM). Además, considero que puede haber más factores que influyan en la venta de helados, como por ejemplo la temperatura media que hace en un día. En este caso, podría ser interesante explorar modelos de series multivariantes como SARIMAX, además de intentar obtener un buen dataset con variables relevantes y probar con modelos de Boosting o Redes Neuronales.

5. Anexo (Código)

```
library(ggplot2)

library(dplyr)

library(zoo)

library(lubridate)

library(tseries)

library(forecast)

library(tidyr)


#-----------------------------------------------------#

# 1º - Lectura de los datos y EDA

#-----------------------------------------------------#


# Leer el dataset

ice_cream <- read.csv("ice_cream.csv", header = TRUE, sep = ",")

ice_cream$DATE <- as.Date(ice_cream$DATE)


# Dividir en train (hasta 2014) y test (2015 en adelante)
```

```
train <- subset(ice_cream, DATE < "2015-01-01")
test <- subset(ice_cream, DATE >= "2015-01-01")

# Dibujamos los datos del conjunto de entrenamiento
ggplot(train, aes(x = DATE, y = IPN31152N)) +
  geom_line(color = "darkblue", size = 1.2) +
  labs(title = "Ventas de Helado a lo Largo del Tiempo (Train)",
       x = "Fecha", y = "Ventas") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# Pintamos ventanas ----------
# Ventana crisis (2007-2009) dentro del conjunto de entrenamiento
train_2008 <- subset(train, DATE >= "2007-01-01" & DATE <= "2009-12-31")
ggplot(train_2008, aes(x = DATE, y = IPN31152N)) +
  geom_line(color = "darkblue", size = 1.2) +
  labs(title = "Ventas de Helado Durante la Crisis de 2008 (Train)",
       x = "Fecha", y = "Ventas") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# Ventana entrada del Euro (2001-2003)
ice_cream_euro <- subset(ice_cream, DATE >= "2001-01-01" & DATE <= "2003-12-31")
ggplot(ice_cream_euro, aes(x = DATE, y = IPN31152N)) +
  geom_line(color = "darkblue", size = 1.2) +
  labs(title = "Ventas de Helado Durante la Entrada del Euro", x = "Fecha", y = "Ventas") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# Media Móvil -----------
# Calcular la media general solo en el conjunto de entrenamiento
media_general <- mean(train$IPN31152N, na.rm = TRUE)
# Calcular la media móvil en el conjunto de entrenamiento
train <- train %>%
  mutate(media_movil = rollmean(IPN31152N, k = 12, fill = NA, align = "right"))
# Reestructurar los datos para graficar
```

```
data_long <- train %>%
  select(DATE, IPN31152N, media_movil) %>%
  pivot_longer(cols = -DATE, names_to = "Serie", values_to = "Ventas")
# Graficar los datos con media móvil y media general
ggplot(data_long, aes(x = DATE, y = Ventas, color = Serie)) +
  geom_line(size = 1.2) +
  geom_hline(aes(yintercept = media_general, color = "Media General"), linetype = "dashed", size = 1.2) +
  scale_color_manual(values = c("IPN31152N" = "lightblue", "media_movil" = "darkgreen", "Media General" = "darkred")) +
  labs(title = "Ventas de Helado con Media General y Media Móvil (Train)",
       x = "Fecha", y = "Ventas", color = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



# Últimos 36 meses ----------
# Filtrar los últimos 36 meses dentro del conjunto de entrenamiento
train_last36 <- train %>%
  filter(DATE >= max(DATE) %m-% months(35))
# Graficar los últimos 36 meses del conjunto de entrenamiento
ggplot(train_last36, aes(x = DATE, y = IPN31152N)) +
  geom_line(color = "darkblue", size = 1.2) +
  scale_x_date(date_labels = "%b %Y", date_breaks = "2 months") +
  labs(title = "Ventas de Helado en los Últimos 36 Meses (Train)",
       x = "Mes", y = "Ventas") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.x = element_text(angle = 45, hjust = 1))


# Descomposiciones ----------
# Descomposición de la serie temporal
ts_train <- ts(train$IPN31152N, start = c(year(min(train$DATE)), month(min(train$DATE))), frequency = 12)


decomp_add <- decompose(ts_train, type = "additive")
plot(decomp_add, col = "darkgreen")
```

```
decomp_mult <- decompose(ts_train, type = "multiplicative")
plot(decomp_mult, col = "darkred")

# ACF y PACF de la serie de entrenamiento
par(mfrow = c(1, 2))
acf(ts_train, main = "ACF - Serie Original", col = "darkgreen")
pacf(ts_train, main = "PACF - Serie Original", col = "darkred")



#-----------------------------------------------------#
# 2º - Modelo AR(p)
#-----------------------------------------------------#
# Modelo AR con datos de entrenamiento
modelo_ar <- Arima(ts_train, order = c(13, 0, 0)) # Porque baja el PACF
summary(modelo_ar)
checkresiduals(modelo_ar)

# Predicciones desde 2015 hasta enero de 2020
h <- nrow(test)
pred_ar <- forecast(modelo_ar, h = h)

# Comparación de predicciones vs test
test$Pred_AR <- as.numeric(pred_ar$mean)

# Unir train y test en un solo dataframe para ggplot
data_plot <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_AR, Serie = "Predicción AR(13)")
)

# Graficar con leyenda
ggplot(data_plot, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
```

```
                    "Test" = "black",
                    "Predicción AR(13)" = "red")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                      "Test" = "solid",
                      "Predicción AR(13)" = "dashed")) +
  labs(title = "Predicciones AR(13) vs Datos Reales",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))


# Crear dataframe con solo test y predicción
data_plot_test <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_AR, Serie = "Predicción AR(13)")
)

# Graficar test y predicción con leyenda
ggplot(data_plot_test, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Test" = "black", "Predicción AR(13)" = "red")) +
  scale_linetype_manual(values = c("Test" = "solid", "Predicción AR(13)" = "dashed")) +
  labs(title = "Predicciones AR(13) vs Datos de Test",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



#Predicción recursiva
# Crear una función para la evaluación progresiva para AR(p)
rolling_forecast_ar <- function(train_data, test_data, order_ar = 13, h = 1) {
  # Inicializar vector para almacenar predicciones
  n_test <- length(test_data)
```

```
 predictions <- numeric(n_test)

 # Convertir datos a series de tiempo
 train_ts <- ts(train_data)

 # Bucle de predicción progresiva
 for (i in 1:n_test) {
  # Ajustar modelo con los datos disponibles
  current_data <- c(train_data, test_data[1:(i-1)])
  current_ts <- ts(current_data)
  modelo <- Arima(current_ts, order = c(order_ar, 0, 0))

  # Hacer predicción para el siguiente paso
  pred <- forecast(modelo, h = 1)
  predictions[i] <- as.numeric(pred$mean)

  # Opcional: mostrar progreso
  if (i %% 10 == 0) {
   cat("Completado:", i, "de", n_test, "\n")
  }
 }

 return(predictions)
}

# Aplicar la función para tu modelo AR(13)
test$Pred_AR_Rolling <- rolling_forecast_ar(train$IPN31152N, test$IPN31152N, order_ar = 13)

# Gráfica 1: Comparación en el período de test
data_plot_test_only <- rbind(
 data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Datos Reales"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_AR, Serie = "Predicción AR(13) Estática"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_AR_Rolling, Serie = "Predicción AR(13) Progresiva")
)

ggplot(data_plot_test_only, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
```

```
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Datos Reales" = "black",
                                "Predicción AR(13) Estática" = "red",
                                "Predicción AR(13) Progresiva" = "green")) +
  scale_linetype_manual(values = c("Datos Reales" = "solid",
                                   "Predicción AR(13) Estática" = "dashed",
                                   "Predicción AR(13) Progresiva" = "dotted")) +
  labs(title = "Comparación de Pronósticos AR(13) en Período de Test",
       subtitle = "Predicción Estática vs Predicción Progresiva",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "bottom") +
  guides(color = guide_legend(nrow = 1))

# Gráfica 2: Errores de predicción
test$Error_Estatico <- test$IPN31152N - test$Pred_AR
test$Error_Progresivo <- test$IPN31152N - test$Pred_AR_Rolling

data_plot_errors <- rbind(
  data.frame(DATE = test$DATE, Error = test$Error_Estatico, Serie = "Error Predicción Estática"),
  data.frame(DATE = test$DATE, Error = test$Error_Progresivo, Serie = "Error Predicción Progresiva")
)

ggplot(data_plot_errors, aes(x = DATE, y = Error, color = Serie)) +
  geom_line(size = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  scale_color_manual(values = c("Error Predicción Estática" = "red",
                                "Error Predicción Progresiva" = "green")) +
  labs(title = "Errores de Predicción AR(13) en Período de Test",
       x = "Fecha", y = "Error (Real - Predicción)", color = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
```

```
    legend.position = "bottom")


# Crear dataframe para visualización de la serie completa con predicciones AR(13)
data_plot_rolling_ar <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_AR, Serie = "Predicción AR(13) Estática"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_AR_Rolling, Serie = "Predicción AR(13) Progresiva")
)

# Graficar comparación de métodos en la serie completa
ggplot(data_plot_rolling_ar, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción AR(13) Estática" = "red",
                                "Predicción AR(13) Progresiva" = "green")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción AR(13) Estática" = "dashed",
                                   "Predicción AR(13) Progresiva" = "dotted")) +
  labs(title = "Comparación de Pronósticos AR(13): Estático vs Progresivo",
       subtitle = "Vista de la Serie Temporal Completa",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))


# Calcular métricas de error para comparar ambos métodos
library(Metrics)
rmse_ar_static <- rmse(test$IPN31152N, test$Pred_AR)
rmse_ar_rolling <- rmse(test$IPN31152N, test$Pred_AR_Rolling)
mae_ar_static <- mae(test$IPN31152N, test$Pred_AR)
mae_ar_rolling <- mae(test$IPN31152N, test$Pred_AR_Rolling)
```

```
# Mostrar resultados
cat("RMSE estático:", rmse_ar_static, "\n")
cat("RMSE progresivo:", rmse_ar_rolling, "\n")
cat("MAE estático:", mae_ar_static, "\n")
cat("MAE progresivo:", mae_ar_rolling, "\n")

library(knitr)

# Crear un dataframe con los resultados
tabla_ar <- data.frame(
  Métrica = c("RMSE estático", "RMSE progresivo", "MAE estático", "MAE progresivo"),
  Valor = c(rmse_ar_static, rmse_ar_rolling, mae_ar_static, mae_ar_rolling)
)

# Imprimir la tabla en formato bonito
kable(tabla_ar, col.names = c("Métrica", "Valor"), caption = "Modelo: AR", digits = 4)
```

```
#-----------------------------------------------------#
# 3º - Modelo MA(q)
#-----------------------------------------------------#
# Ajustar un modelo MA(q) con el q seleccionado
modelo_ma <- Arima(ts_train, order = c(0, 0, 7))  # Tomamos 7, es cuando baja un poco ACF
# Resumen del modelo
summary(modelo_ma)
# Diagnóstico de residuos
checkresiduals(modelo_ma)
```

```
# Predicciones para el horizonte de prueba
h <- nrow(test)
pred_ma <- forecast(modelo_ma, h = h)

# Agregar predicciones al conjunto de test
test$Pred_MA <- as.numeric(pred_ma$mean)



# Unir train, test y predicción MA(q) en un solo dataframe para ggplot
data_plot_ma <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_MA, Serie = "Predicción MA(q)")
)

# Graficar con leyenda
ggplot(data_plot_ma, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción MA(q)" = "red")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción MA(q)" = "dashed")) +
  labs(title = "Predicciones MA(q) vs Datos Reales",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



# Graficar test vs predicción MA(q)
data_plot_test <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_MA, Serie = "Predicción MA(q)")
```

```
)

ggplot(data_plot_test, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Test" = "black", "Predicción MA(q)" = "red")) +
  scale_linetype_manual(values = c("Test" = "solid", "Predicción MA(q)" = "dashed")) +
  labs(title = "Predicciones MA(q) vs Datos de Test",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



# Predicción Recursiva
# Crear una función para la evaluación progresiva
rolling_forecast_ma <- function(train_data, test_data, order_ma = 7, h = 1) {
  # Inicializar vector para almacenar predicciones
  n_test <- length(test_data)
  predictions <- numeric(n_test)

  # Convertir datos a series de tiempo
  train_ts <- ts(train_data)

  # Bucle de predicción progresiva
  for (i in 1:n_test) {
    # Ajustar modelo con los datos disponibles
    current_data <- c(train_data, test_data[1:(i-1)])
    current_ts <- ts(current_data)
    modelo <- Arima(current_ts, order = c(0, 0, order_ma))

    # Hacer predicción para el siguiente paso
    pred <- forecast(modelo, h = 1)
    predictions[i] <- as.numeric(pred$mean)
```

```
  # Opcional: mostrar progreso
  if (i %% 10 == 0) {
   cat("Completado:", i, "de", n_test, "\n")
  }
 }

 return(predictions)
}

# Aplicar la función
test$Pred_MA_Rolling <- rolling_forecast_ma(train$IPN31152N, test$IPN31152N)

# Crear dataframe para visualización
data_plot_rolling <- rbind(
 data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
 data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_MA, Serie = "Predicción MA(q) Estática"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_MA_Rolling, Serie = "Predicción MA(q) Progresiva")
)

# Graficar comparación de métodos
ggplot(data_plot_rolling, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
 geom_line(size = 1.2) +
 scale_color_manual(values = c("Entrenamiento" = "darkblue",
                               "Test" = "black",
                               "Predicción MA(q) Estática" = "red",
                               "Predicción MA(q) Progresiva" = "green")) +
 scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                  "Test" = "solid",
                                  "Predicción MA(q) Estática" = "dashed",
                                  "Predicción MA(q) Progresiva" = "dotted")) +
 labs(title = "Comparación de Pronósticos: Estático vs Progresivo",
      x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
 theme_minimal(base_size = 14) +
 theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

```
# Calcular métricas de error para ambos métodos
library(Metrics)
rmse_ma_static <- rmse(test$IPN31152N, test$Pred_MA)
rmse_ma_rolling <- rmse(test$IPN31152N, test$Pred_MA_Rolling)
mae_ma_static <- mae(test$IPN31152N, test$Pred_MA)
mae_ma_rolling <- mae(test$IPN31152N, test$Pred_MA_Rolling)

# Mostrar resultados
cat("RMSE estático:", rmse_ma_static, "\n")
cat("RMSE progresivo:", rmse_ma_rolling, "\n")
cat("MAE estático:", mae_ma_static, "\n")
cat("MAE progresivo:", mae_ma_rolling, "\n")

# Crear un dataframe con los resultados
tabla_ma <- data.frame(
  Métrica = c("RMSE estático", "RMSE progresivo", "MAE estático", "MAE progresivo"),
  Valor = c(rmse_ma_static, rmse_ma_rolling, mae_ma_static, mae_ma_rolling)
)

# Imprimir la tabla en formato bonito
kable(tabla_ma, col.names = c("Métrica", "Valor"), caption = "Modelo: MA", digits = 4)
```

```
#-------------------------------------------------------#
# 3º - Modelo ARIMA
#-------------------------------------------------------#
# Comprobamos estacionariedad con el Test ADF
adf_result <- adf.test(ts_train)
print(adf_result)  # Si p-value > 0.05 --> No estacionaria

# Hacemos una diferenciación y analizamos si es estacionaria o no
ts_diff1 <- diff(ts_train)  # Primera diferencia
plot(ts_diff1, main = "Serie Diferenciada (Primera Diferencia)", col = "darkblue")

# Comprobamos con el Test ADF la serie diferenciada
adf_test_diff1 <- adf.test(ts_diff1)
print(adf_test_diff1)  # Si p-value < 0.05 --> Estacionaria

# Pintamos ahora ACF y PACF de la serie diferenciada
par(mfrow = c(1, 2))  # Mismo formato para comparación
acf(ts_diff1, main = "ACF - Serie Diferenciada", col = "darkgreen")
pacf(ts_diff1, main = "PACF - Serie Diferenciada", col = "darkred")



# Ajustar un modelo ARIMA(p, d, q) con los valores seleccionados (por ejemplo, p=3, d=1, q=2)
modelo_arima <- Arima(ts_train, order = c(13,1, 4))  # Ajusta p, d, q según ACF/PACF

# Resumen del modelo
summary(modelo_arima)

# Diagnóstico de residuos
checkresiduals(modelo_arima)

# Predicciones para el horizonte de prueba
h <- nrow(test)
pred_arima <- forecast(modelo_arima, h = h)

# Agregar predicciones al conjunto de test
```

```
test$Pred_ARIMA <- as.numeric(pred_arima$mean)


# Unir train, test y predicción ARIMA en un solo dataframe para ggplot
data_plot_arima <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA, Serie = "Predicción ARIMA")
)

# Graficar con leyenda
ggplot(data_plot_arima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción ARIMA" = "red")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción ARIMA" = "dashed")) +
  labs(title = "Predicciones ARIMA vs Datos Reales",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))


# Graficar test vs predicción ARIMA
data_plot_test_arima <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA, Serie = "Predicción ARIMA")
)

ggplot(data_plot_test_arima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Test" = "black", "Predicción ARIMA" = "red")) +
```

```
 scale_linetype_manual(values = c("Test" = "solid", "Predicción ARIMA" = "dashed")) +
 labs(title = "Predicciones ARIMA vs Datos de Test",
      x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
 theme_minimal(base_size = 14) +
 theme(plot.title = element_text(hjust = 0.5, face = "bold"))



# Evaluación progresiva
# Crear función para la evaluación progresiva para ARIMA
# Función de evaluación progresiva robusta para ARIMA
# Función de evaluación progresiva robusta para ARIMA
rolling_forecast_arima_robust <- function(train_data, test_data, order_p = 13, order_d = 1, order_q = 4, h = 1) {
 # Inicializar vector para almacenar predicciones
 n_test <- length(test_data)
 predictions <- numeric(n_test)

 # Convertir datos a series de tiempo
 train_ts <- ts(train_data)

 # Bucle de predicción progresiva
 for (i in 1:n_test) {
  # Ajustar modelo con los datos disponibles
  current_data <- c(train_data, test_data[1:(i-1)])
  current_ts <- ts(current_data)

  # Usar tryCatch para manejar errores
  model_result <- tryCatch({
   # Intentar ajustar el modelo con los parámetros especificados
   modelo <- Arima(current_ts, order = c(order_p, order_d, order_q), method = "ML")

   # Hacer predicción para el siguiente paso
   pred <- forecast(modelo, h = 1)
   list(success = TRUE, prediction = as.numeric(pred$mean))

  }, error = function(e) {
   # Si falla, intentar con un método diferente o parámetros más conservadores
```

```
  cat("Error en iteración", i, ":", e$message, "\n")
  cat("Intentando con método CSS...\n")

  tryCatch({
    # Intento alternativo con método CSS en lugar de ML
    modelo_alt <- Arima(current_ts, order = c(order_p, order_d, order_q), method = "CSS")
    pred_alt <- forecast(modelo_alt, h = 1)
    list(success = TRUE, prediction = as.numeric(pred_alt$mean))

  }, error = function(e2) {
    cat("Error en segundo intento:", e2$message, "\n")
    cat("Intentando con auto.arima...\n")

    # Si ambos fallan, usar auto.arima que es más robusto
    tryCatch({
      modelo_auto <- auto.arima(current_ts, stepwise = TRUE, approximation = TRUE)
      pred_auto <- forecast(modelo_auto, h = 1)
      list(success = TRUE, prediction = as.numeric(pred_auto$mean))

    }, error = function(e3) {
      # Si todo falla, usar un método muy simple (media móvil)
      cat("Todos los métodos fallaron. Usando naive forecast.\n")
      if (i > 1) {
        # Si ya tenemos predicciones anteriores, usar la última
        list(success = TRUE, prediction = predictions[i-1])
      } else {
        # Para la primera predicción, usar el último valor de train
        list(success = TRUE, prediction = tail(train_data, 1))
      }
    })
  })
})

# Guardar la predicción
```

```
  predictions[i] <- model_result$prediction

  # Mostrar progreso
  if (i %% 10 == 0) {
   cat("Completado:", i, "de", n_test, "\n")
  }
 }

 return(predictions)
}

# Aplicar la función para el modelo ARIMA(13,1,4)
test$Pred_ARIMA_Rolling <- rolling_forecast_arima_robust(train$IPN31152N, test$IPN31152N,
                                    order_p = 13, order_d = 1, order_q = 4)


# ----- GRÁFICA 1: SERIE COMPLETA CON PREDICCIONES -----
data_plot_rolling_arima <- rbind(
 data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
 data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA, Serie = "Predicción ARIMA Estática"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA_Rolling, Serie = "Predicción ARIMA Progresiva")
)

ggplot(data_plot_rolling_arima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
 geom_line(size = 1.2) +
 scale_color_manual(values = c("Entrenamiento" = "darkblue",
                "Test" = "black",
                "Predicción ARIMA Estática" = "red",
                "Predicción ARIMA Progresiva" = "green")) +
 scale_linetype_manual(values = c("Entrenamiento" = "solid",
                  "Test" = "solid",
                  "Predicción ARIMA Estática" = "dashed",
                  "Predicción ARIMA Progresiva" = "dotted")) +
 labs(title = "Comparación de Pronósticos ARIMA: Estático vs Progresivo",
    subtitle = "Vista de la Serie Temporal Completa",
```

```
    x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))

# ----- GRÁFICA 2: SOLO PERÍODO DE TEST -----
data_plot_test_only_arima <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Datos Reales"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA, Serie = "Predicción ARIMA Estática"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_ARIMA_Rolling, Serie = "Predicción ARIMA Progresiva")
)

ggplot(data_plot_test_only_arima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Datos Reales" = "black",
                                "Predicción ARIMA Estática" = "red",
                                "Predicción ARIMA Progresiva" = "green")) +
  scale_linetype_manual(values = c("Datos Reales" = "solid",
                                   "Predicción ARIMA Estática" = "dashed",
                                   "Predicción ARIMA Progresiva" = "dotted")) +
  labs(title = "Comparación de Pronósticos ARIMA en Período de Test",
       subtitle = "Predicción Estática vs Predicción Progresiva",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "bottom") +
  guides(color = guide_legend(nrow = 1))

# ----- GRÁFICA 3: ERRORES DE PREDICCIÓN -----
test$Error_Estatico_ARIMA <- test$IPN31152N - test$Pred_ARIMA
test$Error_Progresivo_ARIMA <- test$IPN31152N - test$Pred_ARIMA_Rolling

data_plot_errors_arima <- rbind(
```

```
  data.frame(DATE = test$DATE, Error = test$Error_Estatico_ARIMA, Serie = "Error Predicción Estática"),
  data.frame(DATE = test$DATE, Error = test$Error_Progresivo_ARIMA, Serie = "Error Predicción Progresiva")
)

ggplot(data_plot_errors_arima, aes(x = DATE, y = Error, color = Serie)) +
  geom_line(size = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  scale_color_manual(values = c("Error Predicción Estática" = "red",
                                "Error Predicción Progresiva" = "green")) +
  labs(title = "Errores de Predicción ARIMA en Período de Test",
       x = "Fecha", y = "Error (Real - Predicción)", color = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom")

# ----- MÉTRICAS DE ERROR -----
library(Metrics)
rmse_static_arima <- rmse(test$IPN31152N, test$Pred_ARIMA)
rmse_rolling_arima <- rmse(test$IPN31152N, test$Pred_ARIMA_Rolling)
mae_static_arima <- mae(test$IPN31152N, test$Pred_ARIMA)
mae_rolling_arima <- mae(test$IPN31152N, test$Pred_ARIMA_Rolling)

# Mostrar resultados
cat("RMSE estático (ARIMA):", rmse_static_arima, "\n")
cat("RMSE progresivo (ARIMA):", rmse_rolling_arima, "\n")
cat("MAE estático (ARIMA):", mae_static_arima, "\n")
cat("MAE progresivo (ARIMA):", mae_rolling_arima, "\n")



# Crear un dataframe con los resultados
tabla_arima <- data.frame(
  Métrica = c("RMSE estático", "RMSE progresivo", "MAE estático", "MAE progresivo"),
  Valor = c(rmse_static_arima, rmse_rolling_arima, mae_static_arima, mae_rolling_arima)
)
```

```
# Imprimir la tabla en formato bonito
kable(tabla_arima, col.names = c("Métrica", "Valor"), caption = "Modelo: ARIMA", digits = 4)
```

```
#-------------------------------------------------------#
# 4º - Modelo SARIMA
#-------------------------------------------------------#
# Comprobamos si hay estacionalidad con diferenciación estacional
ts_diff_seasonal <- diff(ts_train, lag = 7)  # Diferenciación estacional
plot(ts_diff_seasonal, main = "Serie Diferenciada Estacionalmente", col = "darkblue")

# Test ADF para la serie diferenciada estacionalmente
adf_test_seasonal <- adf.test(ts_diff_seasonal)
print(adf_test_seasonal) # Sí que es estacionaria

# Pintamos ahora ACF y PACF de la serie diferenciada
par(mfrow = c(2, 1))  # Formato de gráficos


# ACF y PACF de la serie diferenciada estacionalmente
acf(ts_diff_seasonal, main = "ACF - Serie Diferenciada Estacional", col = "darkgreen")
pacf(ts_diff_seasonal, main = "PACF - Serie Diferenciada Estacional", col = "darkred")


# Ajustar un modelo SARIMA(p,d,q)(P,D,Q)[s]
modelo_sarima <- Arima(ts_train, order = c(13,1,4),
                       seasonal = list(order = c(3,0,1), period = 7))
```

```
# Resumen del modelo
summary(modelo_sarima)

# Diagnóstico de residuos
checkresiduals(modelo_sarima)

# Predicciones para el horizonte de prueba
h <- nrow(test)
pred_sarima <- forecast(modelo_sarima, h = h)

# Agregar predicciones al conjunto de test
test$Pred_SARIMA <- as.numeric(pred_sarima$mean)



# Unir train, test y predicción SARIMA en un solo dataframe para ggplot
data_plot_sarima <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA, Serie = "Predicción SARIMA")
)

# Graficar con leyenda
ggplot(data_plot_sarima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción SARIMA" = "red")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción SARIMA" = "dashed")) +
  labs(title = "Predicciones SARIMA vs Datos Reales",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```

```
# Graficar test vs predicción SARIMA
data_plot_test_sarima <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA, Serie = "Predicción SARIMA")
)

ggplot(data_plot_test_sarima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Test" = "black", "Predicción SARIMA" = "red")) +
  scale_linetype_manual(values = c("Test" = "solid", "Predicción SARIMA" = "dashed")) +
  labs(title = "Predicciones SARIMA vs Datos de Test",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))



# Evaluación progresiva
# Crear función para la evaluación progresiva para SARIMA
# Función de evaluación progresiva robusta para SARIMA
rolling_forecast_sarima_robust <- function(train_data, test_data, order_p = 13, order_d = 1, order_q = 4,
                                           seasonal_P = 3, seasonal_D = 0, seasonal_Q = 1, s = 7, h = 1) {
  # Inicializar vector para almacenar predicciones
  n_test <- length(test_data)
  predictions <- numeric(n_test)

  # Convertir datos a series de tiempo
  train_ts <- ts(train_data)

  # Bucle de predicción progresiva
  for (i in 1:n_test) {
    # Ajustar modelo con los datos disponibles
    current_data <- c(train_data, test_data[1:(i-1)])
    current_ts <- ts(current_data)
```

```
# Usar tryCatch para manejar errores
model_result <- tryCatch({
 # Intentar ajustar el modelo con los parámetros especificados
 modelo <- Arima(current_ts,
                 order = c(order_p, order_d, order_q),
                 seasonal = list(order = c(seasonal_P, seasonal_D, seasonal_Q), period = s),
                 method = "ML")

 # Hacer predicción para el siguiente paso
 pred <- forecast(modelo, h = 1)
 list(success = TRUE, prediction = as.numeric(pred$mean))

}, error = function(e) {
 # Si falla, intentar con un método diferente o parámetros más conservadores
 cat("Error en iteración", i, ":", e$message, "\n")
 cat("Intentando con método CSS...\n")

 tryCatch({
  # Intento alternativo con método CSS
  modelo_alt <- Arima(current_ts,
                      order = c(order_p, order_d, order_q),
                      seasonal = list(order = c(seasonal_P, seasonal_D, seasonal_Q), period = s),
                      method = "CSS")
  pred_alt <- forecast(modelo_alt, h = 1)
  list(success = TRUE, prediction = as.numeric(pred_alt$mean))

 }, error = function(e2) {
  cat("Error en segundo intento:", e2$message, "\n")
  cat("Intentando con auto.arima...\n")

  # Si ambos fallan, usar auto.arima que es más robusto
  tryCatch({
   modelo_auto <- auto.arima(current_ts, seasonal = TRUE, stepwise = TRUE, approximation = TRUE)
   pred_auto <- forecast(modelo_auto, h = 1)
   list(success = TRUE, prediction = as.numeric(pred_auto$mean))
```

```
      }, error = function(e3) {
        # Si todo falla, usar un método muy simple (media móvil o naive)
        cat("Todos los métodos fallaron. Usando naive forecast.\n")
        if (i > 1) {
          # Si ya tenemos predicciones anteriores, usar la última
          list(success = TRUE, prediction = predictions[i-1])
        } else {
          # Para la primera predicción, usar el último valor de train
          list(success = TRUE, prediction = tail(train_data, 1))
        }
      })
    })
  })

  # Guardar la predicción
  predictions[i] <- model_result$prediction

  # Mostrar progreso
  if (i %% 10 == 0) {
    cat("Completado:", i, "de", n_test, "\n")
  }
 }

 return(predictions)
}

# Aplicar la función para el modelo SARIMA(13,1,4)(3,0,1)[7]
test$Pred_SARIMA_Rolling <- rolling_forecast_sarima_robust(train$IPN31152N, test$IPN31152N,
                                    order_p = 13, order_d = 1, order_q = 4,
                                    seasonal_P = 3, seasonal_D = 0, seasonal_Q = 1, s = 7)

# ----- GRÁFICA 1: SERIE COMPLETA CON PREDICCIONES -----
data_plot_rolling_sarima <- rbind(
```

```
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA, Serie = "Predicción SARIMA Estática"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Rolling, Serie = "Predicción SARIMA Progresiva")
)

ggplot(data_plot_rolling_sarima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción SARIMA Estática" = "red",
                                "Predicción SARIMA Progresiva" = "green")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción SARIMA Estática" = "dashed",
                                   "Predicción SARIMA Progresiva" = "dotted")) +
  labs(title = "Predicciones SARIMA Estática vs Progresiva",
       subtitle = "Vista de la Serie Temporal Completa",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))



# ----- GRÁFICA 2: SOLO PERÍODO DE TEST -----
data_plot_test_only_sarima <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Datos Reales"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA, Serie = "Predicción SARIMA Estática"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Rolling, Serie = "Predicción SARIMA Progresiva")
)

ggplot(data_plot_test_only_sarima, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Datos Reales" = "black",
                                "Predicción SARIMA Estática" = "red",
```

```
                      "Predicción SARIMA Progresiva" = "green")) +
  scale_linetype_manual(values = c("Datos Reales" = "solid",
                                   "Predicción SARIMA Estática" = "dashed",
                                   "Predicción SARIMA Progresiva" = "dotted")) +
  labs(title = "Predicciones SARIMA Estática vs Progresiva en Período de Test",
       subtitle = "Comparación de Predicción Estática y Progresiva",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5),
        legend.position = "bottom") +
  guides(color = guide_legend(nrow = 1))

# ----- GRÁFICA 3: ERRORES DE PREDICCIÓN -----
test$Error_Estatico_SARIMA <- test$IPN31152N - test$Pred_SARIMA
test$Error_Progresivo_SARIMA <- test$IPN31152N - test$Pred_SARIMA_Rolling

data_plot_errors_sarima <- rbind(
  data.frame(DATE = test$DATE, Error = test$Error_Estatico_SARIMA, Serie = "Error Predicción Estática"),
  data.frame(DATE = test$DATE, Error = test$Error_Progresivo_SARIMA, Serie = "Error Predicción Progresiva")
)

ggplot(data_plot_errors_sarima, aes(x = DATE, y = Error, color = Serie)) +
  geom_line(size = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "gray50") +
  scale_color_manual(values = c("Error Predicción Estática" = "red",
                                "Error Predicción Progresiva" = "green")) +
  labs(title = "Errores de Predicción SARIMA en Período de Test",
       x = "Fecha", y = "Error (Real - Predicción)", color = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        legend.position = "bottom")
```

```
# ----- MÉTRICAS DE ERROR -----

library(Metrics)

rmse_static_sarima <- rmse(test$IPN31152N, test$Pred_SARIMA)

rmse_rolling_sarima <- rmse(test$IPN31152N, test$Pred_SARIMA_Rolling)

mae_static_sarima <- mae(test$IPN31152N, test$Pred_SARIMA)

mae_rolling_sarima <- mae(test$IPN31152N, test$Pred_SARIMA_Rolling)


# Mostrar resultados

cat("RMSE estático (SARIMA):", rmse_static_sarima, "\n")

cat("RMSE progresivo (SARIMA):", rmse_rolling_sarima, "\n")

cat("MAE estático (SARIMA):", mae_static_sarima, "\n")

cat("MAE progresivo (SARIMA):", mae_rolling_sarima, "\n")


# Crear un dataframe con los resultados

tabla_sarima <- data.frame(

  Métrica = c("RMSE estático", "RMSE progresivo", "MAE estático", "MAE progresivo"),

  Valor = c(rmse_static_sarima, rmse_rolling_sarima, mae_static_sarima, mae_rolling_sarima)

)


# Imprimir la tabla en formato bonito

kable(tabla_sarima, col.names = c("Métrica", "Valor"), caption = "Modelo: SARIMA", digits = 4)









#-----------------------------------------------------#

# 5º - MODELO AUTOMÁTICO

#-----------------------------------------------------#


library(forecast)


modelo_sarima_auto <- auto.arima(ts_train, seasonal = TRUE)  # Ajustar modelo SARIMA

summary(modelo_sarima_auto)
```

```
# ---- DIAGNÓSTICO DE RESIDUOS ----
checkresiduals(modelo_sarima_auto)

# ---- PREDICCIONES ESTÁTICAS ----
h <- nrow(test)
pred_sarima_auto <- forecast(modelo_sarima_auto, h = h)

# Agregar predicciones al dataset de test
test$Pred_SARIMA_Auto <- as.numeric(pred_sarima_auto$mean)

# ---- GRÁFICO: SERIE TEMPORAL CON PREDICCIONES ----
data_plot_sarima_auto <- rbind(
  data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Auto, Serie = "Predicción AUTO SARIMA")
)

ggplot(data_plot_sarima_auto, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Entrenamiento" = "darkblue",
                                "Test" = "black",
                                "Predicción AUTO SARIMA" = "red")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                                   "Test" = "solid",
                                   "Predicción AUTO SARIMA" = "dashed")) +
  labs(title = "Predicción AUTO SARIMA vs Datos Reales",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# ---- EVALUACIÓN PROGRESIVA ----
rolling_forecast_sarima_auto <- function(train_data, test_data, modelo_auto, h = 1) {
  n_test <- length(test_data)
```

```
 predictions <- numeric(n_test)

 for (i in 1:n_test) {
  current_data <- c(train_data, test_data[1:(i-1)])
  current_ts <- ts(current_data)

  modelo_result <- tryCatch({
   modelo <- auto.arima(current_ts, seasonal = TRUE, stepwise = TRUE, approximation = TRUE)
   pred <- forecast(modelo, h = 1)
   list(success = TRUE, prediction = as.numeric(pred$mean))
  }, error = function(e) {
   cat("Error en iteración", i, ": usando naive forecast.\n")
   list(success = TRUE, prediction = ifelse(i > 1, predictions[i-1], tail(train_data, 1)))
  })

  predictions[i] <- modelo_result$prediction
 }

 return(predictions)
}

# Aplicar evaluación progresiva
test$Pred_SARIMA_Auto_Rolling <- rolling_forecast_sarima_auto(train$IPN31152N, test$IPN31152N, modelo_sarima_auto)

# ---- GRÁFICO: PREDICCIÓN ESTÁTICA VS PROGRESIVA ----
data_plot_rolling_sarima_auto <- rbind(
 data.frame(DATE = train$DATE, Ventas = train$IPN31152N, Serie = "Entrenamiento"),
 data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Auto, Serie = "Predicción AUTO SARIMA Estática"),
 data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Auto_Rolling, Serie = "Predicción AUTO SARIMA Progresiva")
)

ggplot(data_plot_rolling_sarima_auto, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
 geom_line(size = 1.2) +
 scale_color_manual(values = c("Entrenamiento" = "darkblue",
                               "Test" = "black",
```

```
                    "Predicción AUTO SARIMA Estática" = "red",
                    "Predicción AUTO SARIMA Progresiva" = "green")) +
  scale_linetype_manual(values = c("Entrenamiento" = "solid",
                      "Test" = "solid",
                      "Predicción AUTO SARIMA Estática" = "dashed",
                      "Predicción AUTO SARIMA Progresiva" = "dotted")) +
  labs(title = "Predicción AUTO SARIMA Estática vs Progresiva",
       subtitle = "Vista Completa de la Serie Temporal",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

# Graficar test vs predicción AUTO SARIMA (Estática y Progresiva) con zoom en el período de test
data_plot_rolling_sarima_auto_test <- rbind(
  data.frame(DATE = test$DATE, Ventas = test$IPN31152N, Serie = "Test"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Auto, Serie = "Predicción AUTO SARIMA Estática"),
  data.frame(DATE = test$DATE, Ventas = test$Pred_SARIMA_Auto_Rolling, Serie = "Predicción AUTO SARIMA Progresiva")
)

ggplot(data_plot_rolling_sarima_auto_test, aes(x = DATE, y = Ventas, color = Serie, linetype = Serie)) +
  geom_line(size = 1.2) +
  scale_color_manual(values = c("Test" = "black",
                    "Predicción AUTO SARIMA Estática" = "red",
                    "Predicción AUTO SARIMA Progresiva" = "green")) +
  scale_linetype_manual(values = c("Test" = "solid",
                      "Predicción AUTO SARIMA Estática" = "dashed",
                      "Predicción AUTO SARIMA Progresiva" = "dotted")) +
  labs(title = "Predicción AUTO SARIMA Estática vs Progresiva",
       subtitle = "Zoom en Período de Test",
       x = "Fecha", y = "Ventas", color = "Serie", linetype = "Serie") +
  theme_minimal(base_size = 14) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5))
```

```
# ---- MÉTRICAS DE ERROR ----

rmse_static_sarima_auto <- rmse(test$IPN31152N, test$Pred_SARIMA_Auto)

rmse_rolling_sarima_auto <- rmse(test$IPN31152N, test$Pred_SARIMA_Auto_Rolling)

mae_static_sarima_auto <- mae(test$IPN31152N, test$Pred_SARIMA_Auto)

mae_rolling_sarima_auto <- mae(test$IPN31152N, test$Pred_SARIMA_Auto_Rolling)


tabla_sarima_auto <- data.frame(

 Métrica = c("RMSE Estático", "RMSE Progresivo", "MAE Estático", "MAE Progresivo"),

 Valor = c(rmse_static_sarima_auto, rmse_rolling_sarima_auto, mae_static_sarima_auto, mae_rolling_sarima_auto)

)


kable(tabla_sarima_auto, col.names = c("Métrica", "Valor"), caption = "Modelo: AUTO SARIMA", digits = 4)




#-----------------------------------------------------#

# 6º - COMPARACIÓN DE MODELOS

#-----------------------------------------------------#

library(dplyr)

library(knitr)


# Crear dataframe con los resultados

resultados <- data.frame(

 Modelo = c("AR", "MA", "ARIMA", "SARIMA", "AUTO SARIMA"),

 AIC = c(AIC(modelo_ar), AIC(modelo_ma), AIC(modelo_arima), AIC(modelo_sarima), AIC(modelo_sarima_auto)),

 RMSE_Train = c(modelo_ar$sigma2^0.5, modelo_ma$sigma2^0.5, modelo_arima$sigma2^0.5, modelo_sarima$sigma2^0.5,
modelo_sarima_auto$sigma2^0.5),

 RMSE_Test_Static = c(rmse_ar_static, rmse_ma_static, rmse_static_arima, rmse_static_sarima, rmse_static_sarima_auto),

 RMSE_Test_Rolling = c(rmse_ar_rolling, rmse_ma_rolling, rmse_rolling_arima, rmse_rolling_sarima,
rmse_rolling_sarima_auto),

 MAE_Test_Static = c(mae_ar_static, mae_ma_static, mae_static_arima, mae_static_sarima, mae_static_sarima_auto),

 MAE_Test_Rolling = c(mae_ar_rolling, mae_ma_rolling, mae_rolling_arima, mae_rolling_sarima, mae_rolling_sarima_auto)

)


# Mostrar tabla en formato bonito

kable(resultados, digits = 4, caption = "Comparación de Modelos ARIMA")
```