1 Carga dataset

In [1]:
```
install.packages("xgboost")
install.packages("Metrics")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

In [3]:
```
library(Metrics)
library(readxl)
library(dplyr)
library(lubridate)
library(forecast)
library(zoo)  # <- REQUIRED for na.locf()
library(ggplot2)
library(xgboost)
library(tidyr)
library(tseries)

# Read csv file t10305c1.csv
raw_df <- read.csv("/workspace/dada_notebooks/Trabajos_no_TFM/t10305c1.csv", header
# get only complete cases
raw_df <- raw_df[complete.cases(raw_df), ]

head(raw_df)
```

A data.frame: 6 × 4

	X	Total	Tourists	Day.trippers
	<chr>	<dbl>	<dbl>	<dbl>
1	03/2025 (p)	1968.7	1370.3	598.5
2	02/2025 (p)	1704.5	1136.6	567.9
3	01/2025 (p)	1607.3	1076.8	530.5
4	Dec-24	1611.8	1068.8	542.9
5	Nov-24	1949.7	1262.5	687.2
6	Oct-24	2751.6	1801.8	949.9

2. EDA

2.1 Convertir columna X a formato Date.

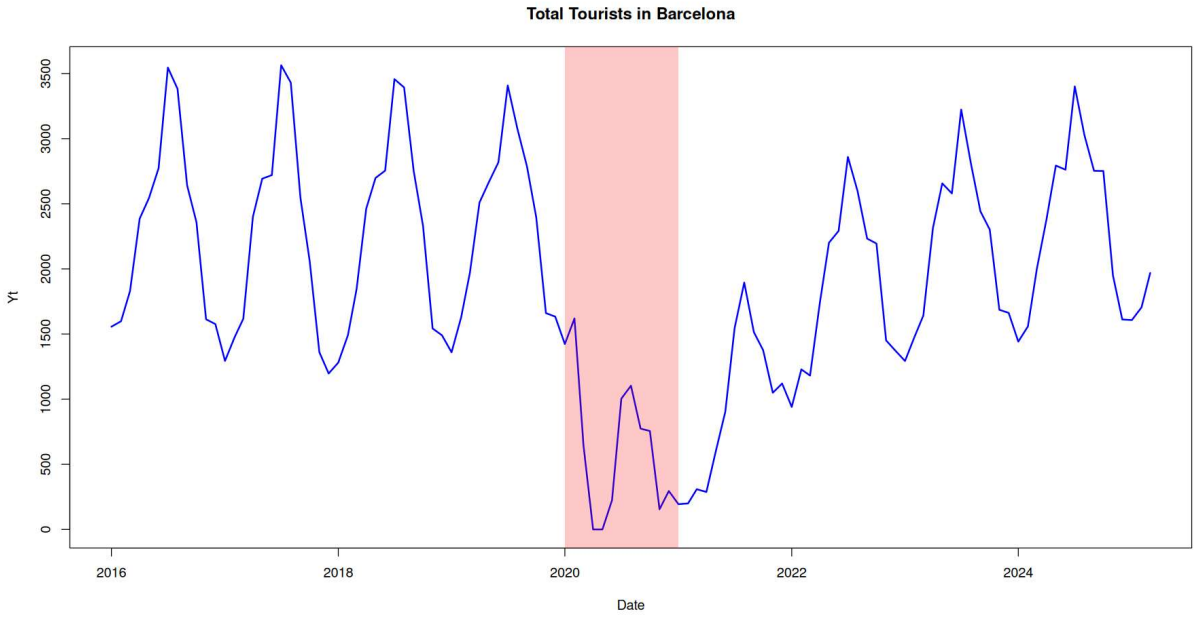
In [4]:
```
df <- raw_df %>%
  mutate(
    CleanDate = case_when(
      grepl("\\d{2}/\\d{4}", .data$X) ~ dmy(paste0("01/", gsub(" \\(p\\)", "", .dat
      TRUE ~ my(.data$X)
    )
  )

head(df)
```

A data.frame: 6 × 5

	X	Total	Tourists	Day.trippers	CleanDate
	<chr>	<dbl>	<dbl>	<dbl>	<date>
1	03/2025 (p)	1968.7	1370.3	598.5	2025-03-01
2	02/2025 (p)	1704.5	1136.6	567.9	2025-02-01
3	01/2025 (p)	1607.3	1076.8	530.5	2025-01-01
4	Dec-24	1611.8	1068.8	542.9	2024-12-01
5	Nov-24	1949.7	1262.5	687.2	2024-11-01
6	Oct-24	2751.6	1801.8	949.9	2024-10-01

2.2 Visualización de la serie temporal

In [5]:
```
# Set white background
options(repr.plot.width = 15, repr.plot.height = 8)
par(bg = "white")

# Plot the time series
plot(df$CleanDate, df$Total, type = "l", col = "blue", lwd = 2,
     xlab = "Date", ylab = "Yt", main = "Total Tourists in Barcelona")


# Add red opaque section from 2020 to 2021
rect(xleft = as.Date("2020-01-01"), xright = as.Date("2021-01-01"),
     ybottom = par("usr")[3], ytop = par("usr")[4],
     col = rgb(1, 0, 0, alpha = 0.2), border = NA)
```

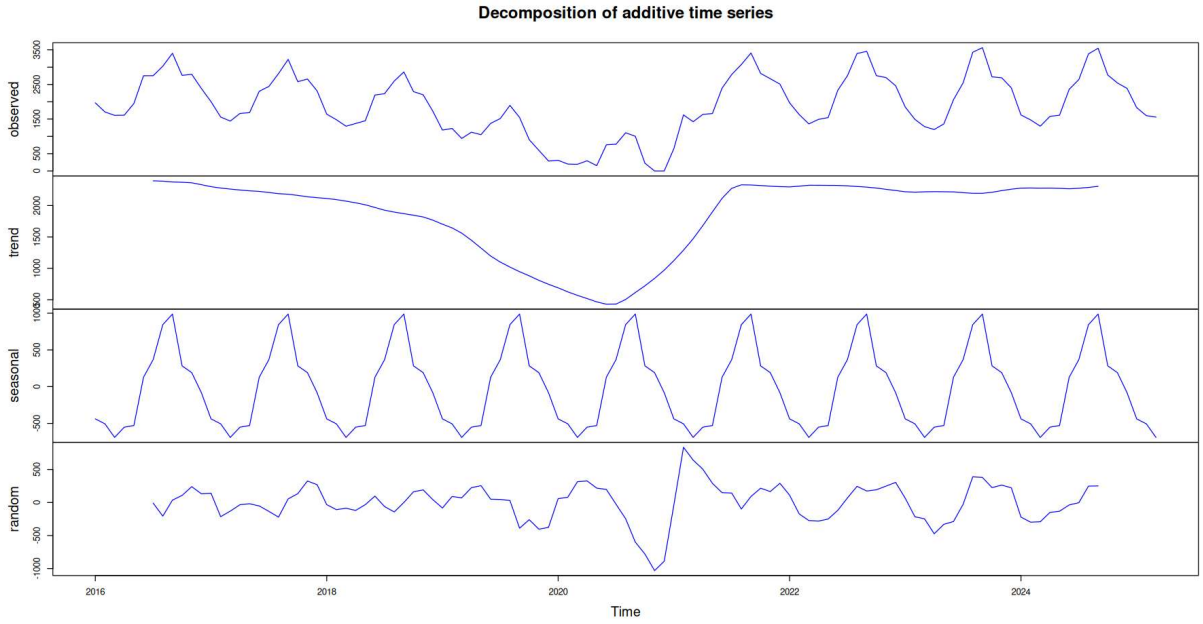


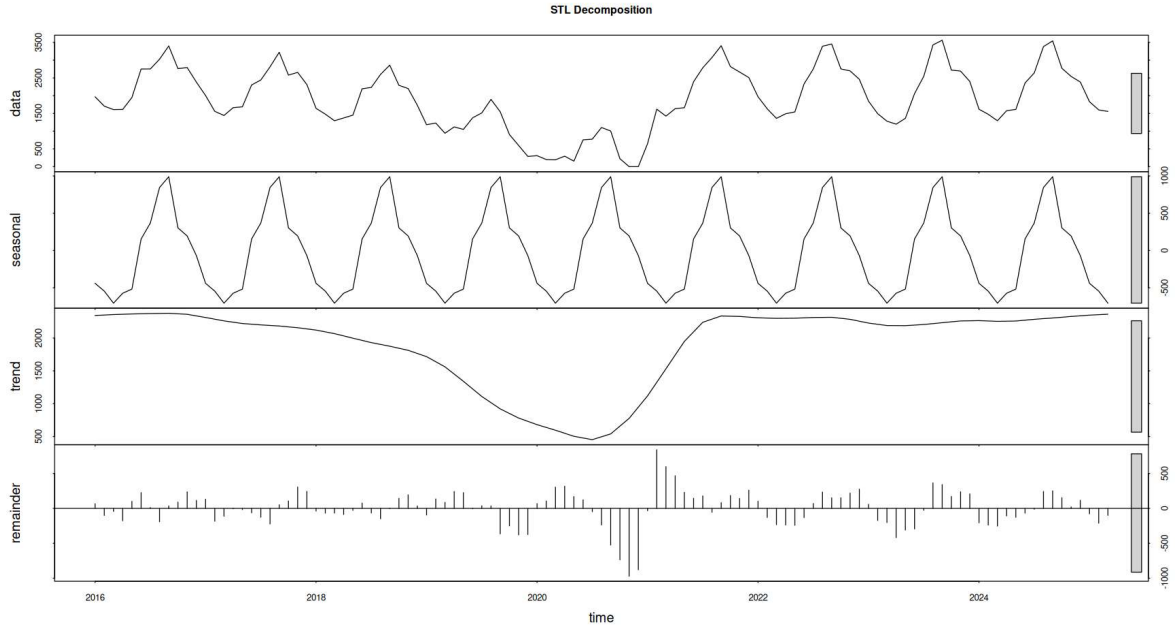
In [6]:
```
sum(is.na(df$CleanDate))       # Count NAs
head(df$CleanDate[is.na(df$CleanDate)])  # View malformed entries
```

0

2.3 Descomposición de componentes

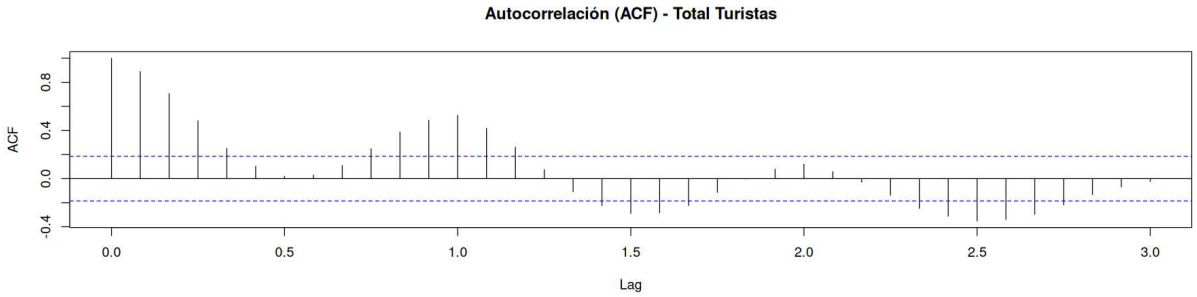
In [7]:
```
# Crear objeto ts
ts_total <- ts(df$Total, start = c(year(min(df$CleanDate)), month(min(df$CleanDate)
par(mfrow = c(2, 1), bg = "white")
# Descomposición clásica aditiva
decomp <- decompose(ts_total, type = "additive")
plot(decomp, col = "blue")

# Alternativamente, descomposición STL
stl_decomp <- stl(ts_total, s.window = "periodic")
plot(stl_decomp, main = "STL Decomposition")
```

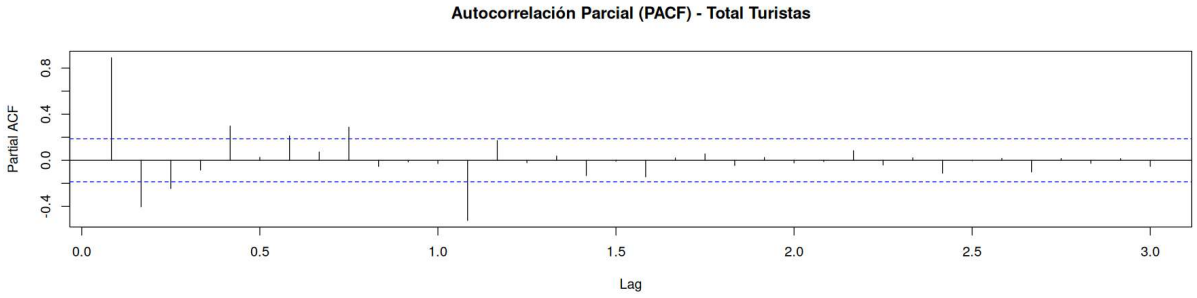




2.4 Autocorrelación (ACF) y autocorrelación parcial (PACF)

In [8]:
```r
# ACF: mide correlación entre valores y sus retardos
par(mfrow = c(2, 1), bg = "white")
acf(ts_total, lag.max = 36, main = "Autocorrelación (ACF) - Total Turistas")

# PACF: mide la correlación neta, eliminando efectos intermedios
pacf(ts_total, lag.max = 36, main = "Autocorrelación Parcial (PACF) - Total Turista
```

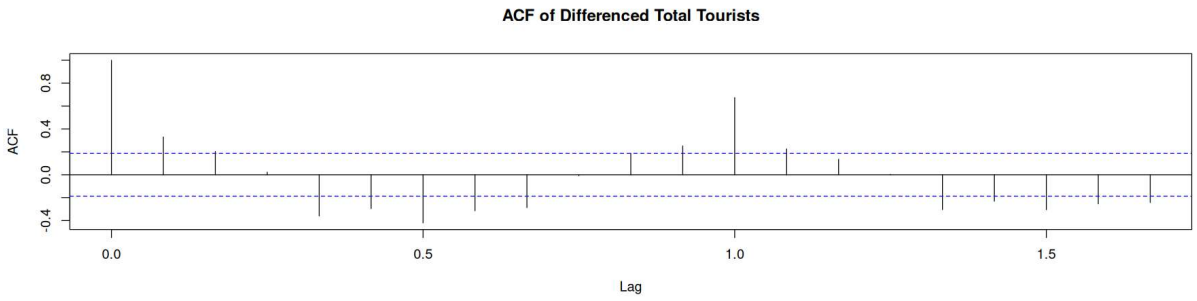


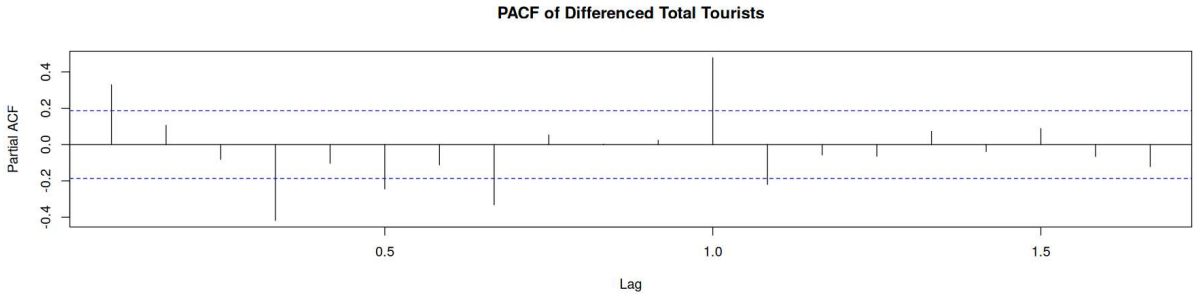


2.5 Estacionariedad: Diferenciación y ACF/PACF sobre serie diferenciada

In [9]:
```r
# Test de Dickey-Fuller aumentado sobre la serie original
adf_result <- adf.test(ts_total)
print(adf_result)

# Interpretación automática del p-value
if (adf_result$p.value < 0.05) {
  cat("\nResultado: La serie es ESTACIONARIA (p-value =", round(adf_result$p.value,
} else {
  cat("\nResultado: La serie NO es estacionaria (p-value =", round(adf_result$p.val
}

# Primera diferenciación
ts_diff <- diff(ts_total)

# ACF and PACF plots for differenced data
par(mfrow = c(2, 1), bg = "white")  # Set up plotting area for two plots
acf(ts_diff, main = "ACF of Differenced Total Tourists")
pacf(ts_diff, main = "PACF of Differenced Total Tourists")
```

```
        Augmented Dickey-Fuller Test

data:  ts_total
Dickey-Fuller = -2.9398, Lag order = 4, p-value = 0.1866
alternative hypothesis: stationary


Resultado: La serie NO es estacionaria (p-value = 0.1866 )
```





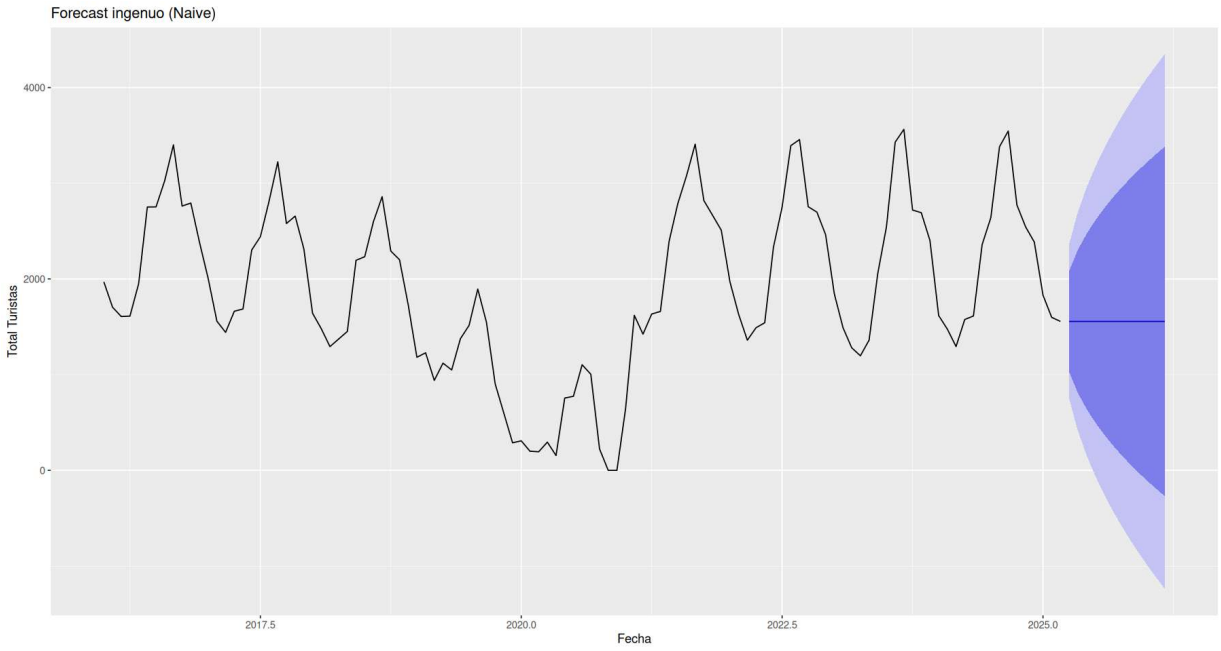
3. División en periodo muestral y extramuestral (80/20)

In [10]:
```
# Split data
train_df <- df %>% filter(year(CleanDate) < 2024)
test_df <- df %>% filter(year(CleanDate) == 2025)
test_df <- test_df %>% arrange(CleanDate)
```

4. Modelos deterministas

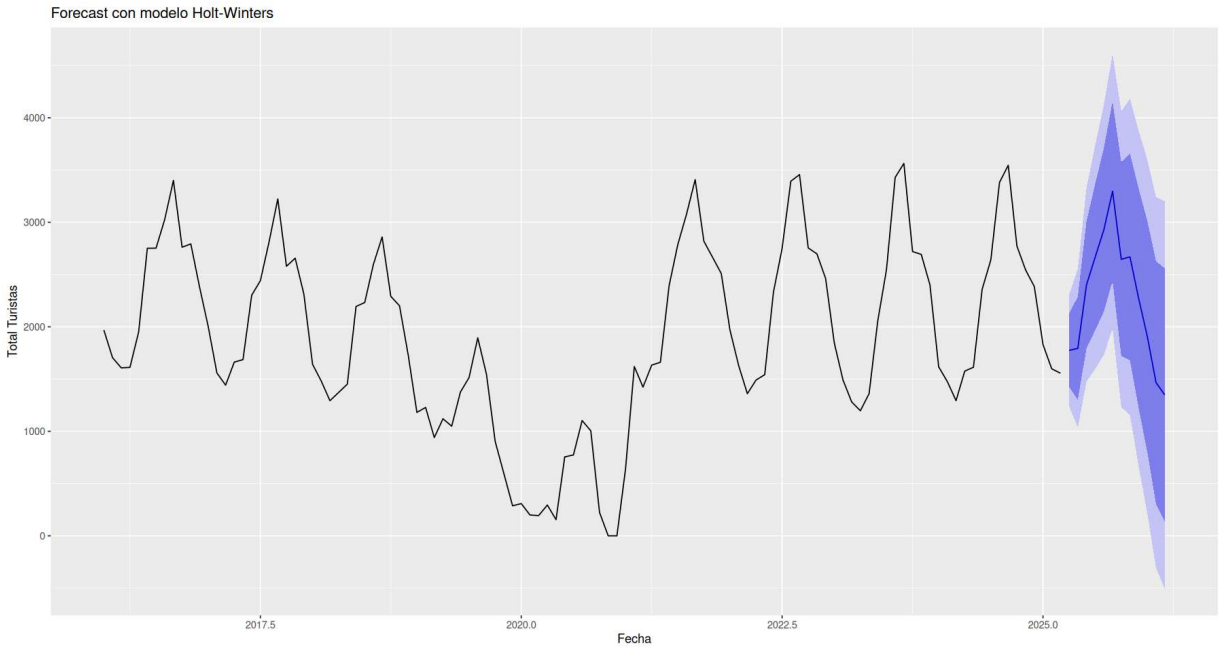
4.1 Modelo ingenuo (Naive)

In [11]:
```
# Crear serie temporal completa
ts_total <- ts(df$Total, start = c(year(min(df$CleanDate)), month(min(df$CleanDate)

# Ajustar modelo ingenuo con última observación
naive_model <- naive(ts_total, h = 12)

# Visualizar predicción
autoplot(naive_model) +
  ggtitle("Forecast ingenuo (Naive)") +
  ylab("Total Turistas") +
  xlab("Fecha")
```



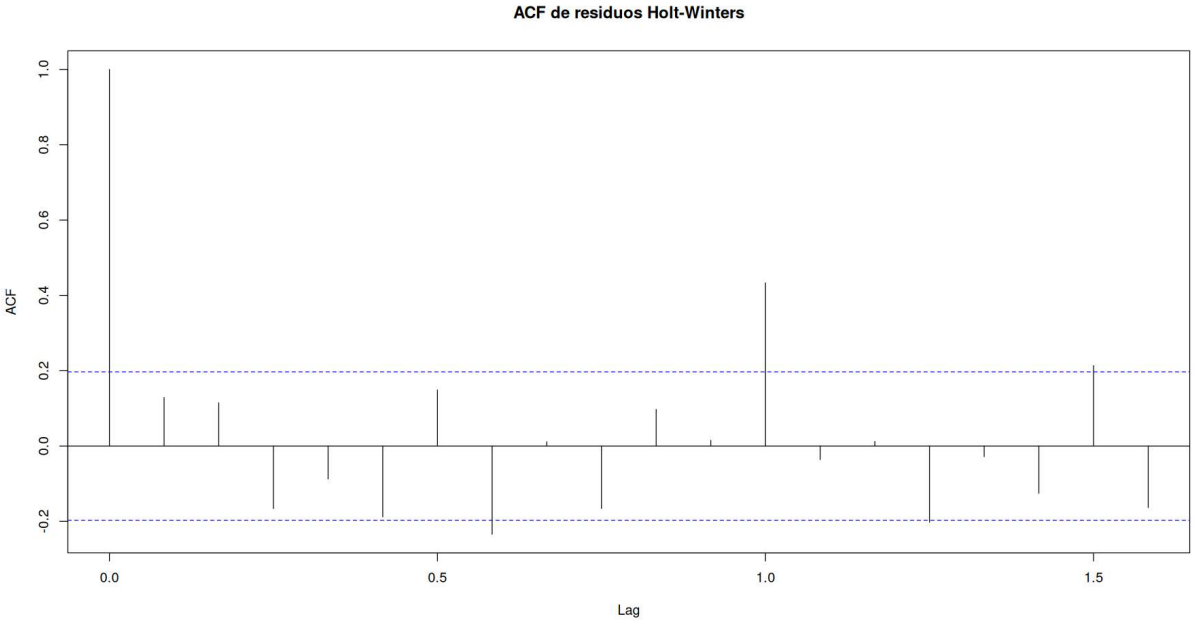
4.2 Modelo de Holt-Winters (suavizado exponencial triple)

In [12]:
```r
# Ajustar modelo Holt-Winters sobre la serie completa
hw_model <- HoltWinters(ts_total)

# Predecir los próximos 12 meses
hw_forecast <- forecast(hw_model, h = 12)

# Visualización del pronóstico
autoplot(hw_forecast) +
  ggtitle("Forecast con modelo Holt-Winters") +
  ylab("Total Turistas") +
  xlab("Fecha")
```



4.2.1 Residuos Holt-Winters

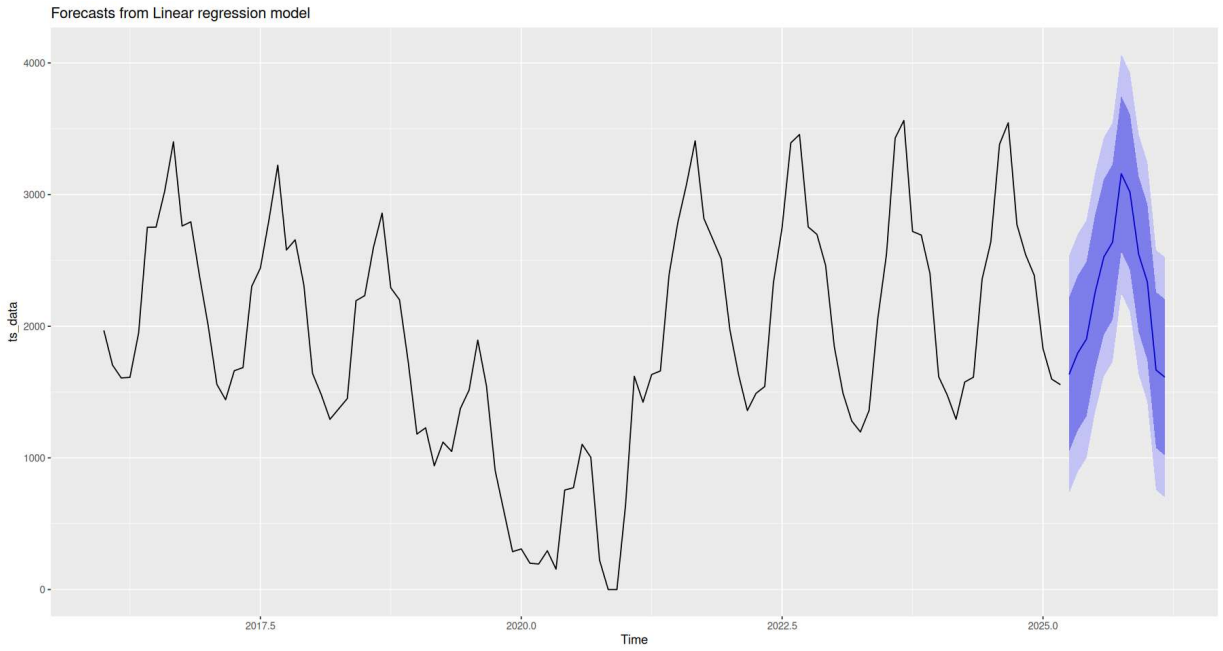
In [13]:
```
par(mfrow = c(1, 1), bg = "white")
acf(residuals(hw_model), main = "ACF de residuos Holt-Winters")
```



4.3 Modelo de regresión lineal con tendencia, estacionalidad y dummy COVID (TSLM)

In [14]:
```
df <- df %>%
  mutate(
    Date = as.Date(CleanDate),
    covid_dummy = if_else(year(Date) == 2020 | (year(Date) == 2021 & month(Date) <=
    trend = row_number(),
```

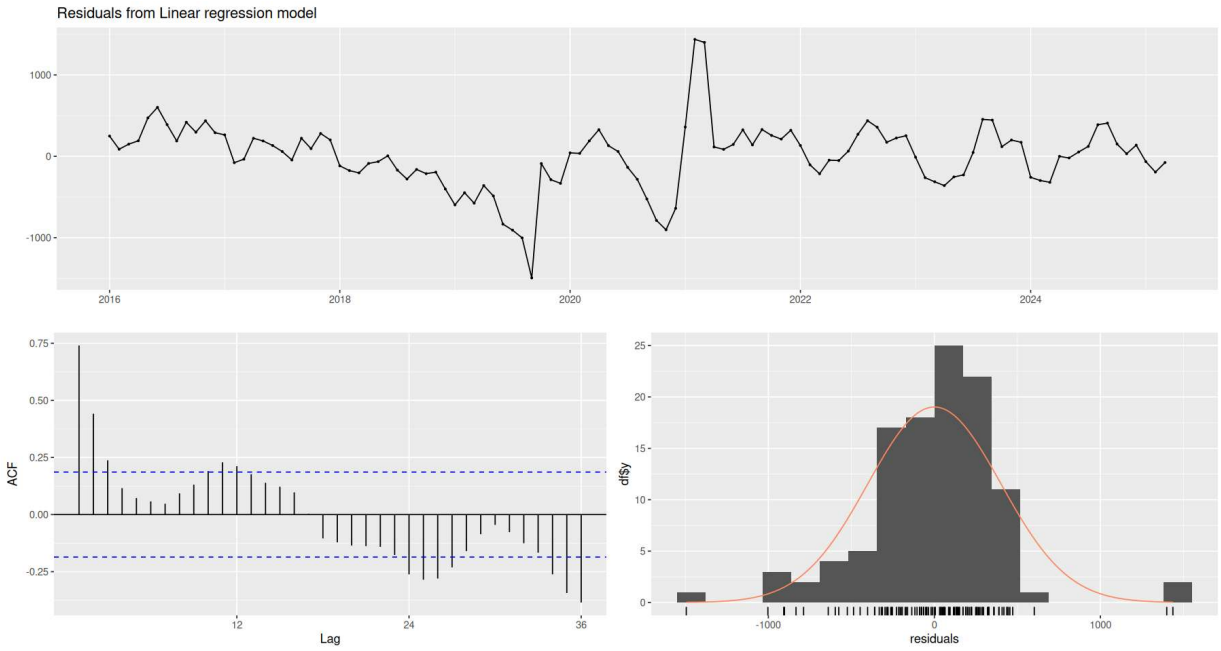
```
    season = factor(month(CleanDate))  # <- se agregó esta línea para definir 'seas
  ) %>%
  filter(!is.na(Date), !is.na(Total))

ts_data <- ts(df$Total, start = c(year(min(df$Date)), month(min(df$Date))), frequen

# Step 3: Fit model using tslm with data= parameter
fit <- tslm(ts_data ~ trend + season + covid_dummy, data = df)

# Step 4: Create newdata for forecast
future_dates <- seq(max(df$Date) %m+% months(1), by = "month", length.out = 12)

newdata <- data.frame(
  trend = (nrow(df) + 1):(nrow(df) + 12),
  season = factor(rep(1:12, length.out = 12), levels = levels(df$season)),
  covid_dummy = rep(0, 12)  # no COVID effect assumed
)

# Step 5: Forecast and plot
fc <- forecast(fit, newdata = newdata)

autoplot(fc)
```



4.3.1 Residuos TSLM

In [15]:
```
checkresiduals(fit)
```

```
	Breusch-Godfrey test for serial correlation of order up to 22

data:  Residuals from Linear regression model
LM test = 73.405, df = 22, p-value = 1.906e-07
```



5. Modelos estocásticos (ARIMA / SARIMA)

5.1 Identificación del modelo: necesidad de diferenciación y análisis ACF/PACF

El test de Dickey-Fuller aplicado a la serie original (ts_total) devuelve un p-value = 0.1866, por lo tanto no se puede rechazar la hipótesis nula de que la serie tiene una raíz unitaria. Esto indica que la serie no es estacionaria y, por tanto, requiere diferenciación.

Tras aplicar la primera diferencia, se observan los siguientes patrones:

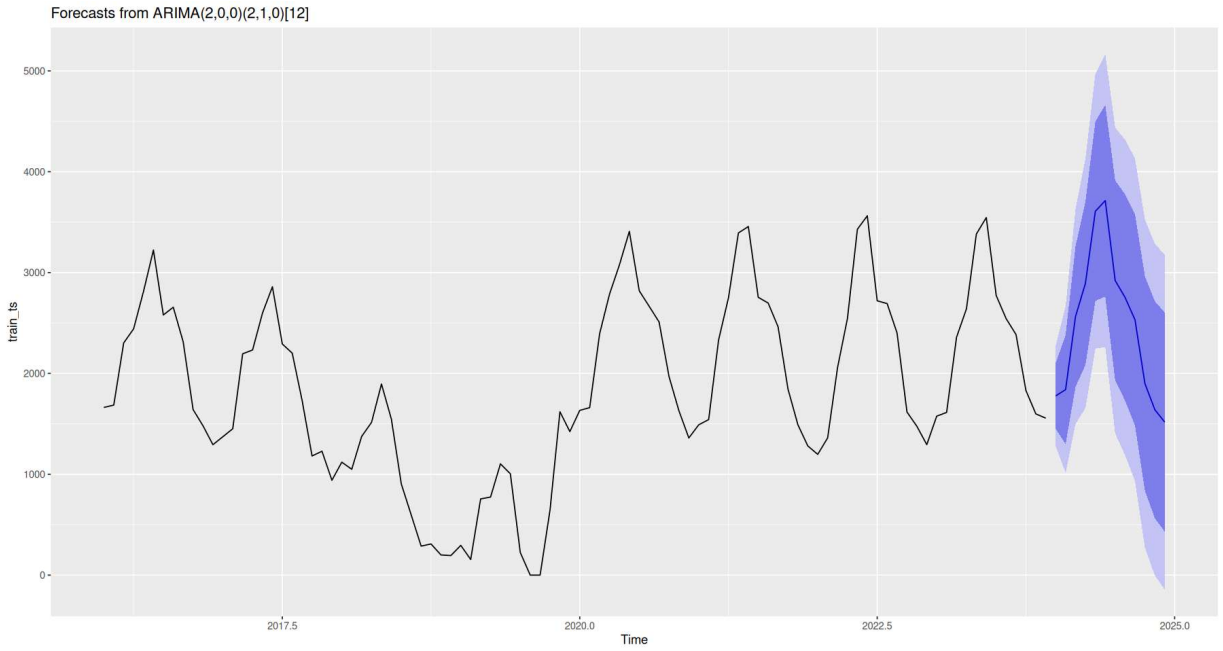
- En la ACF, hay un pico fuerte en el primer rezago y luego decae, indicando que la componente MA (q) puede estar presente.
- En la PACF, hay un pico claro en el primer rezago, lo cual sugiere que la componente AR (p) también puede estar presente.

Con base en este análisis, se considera razonable ajustar un modelo ARIMA(p=1, d=1, q=1) o permitir que auto.arima() seleccione automáticamente los parámetros óptimos.

5.2 Estimación del modelo SARIMA (auto.arima)

In [16]:
```
# Fit SARIMA model
# turn to ts
train_df <- train_df %>%
  mutate(CleanDate = as.Date(CleanDate))
train_ts <- ts(train_df$Total, start = c(year(min(train_df$CleanDate)), month(min(t
sarima_fit <- auto.arima(train_ts, seasonal = TRUE)
```

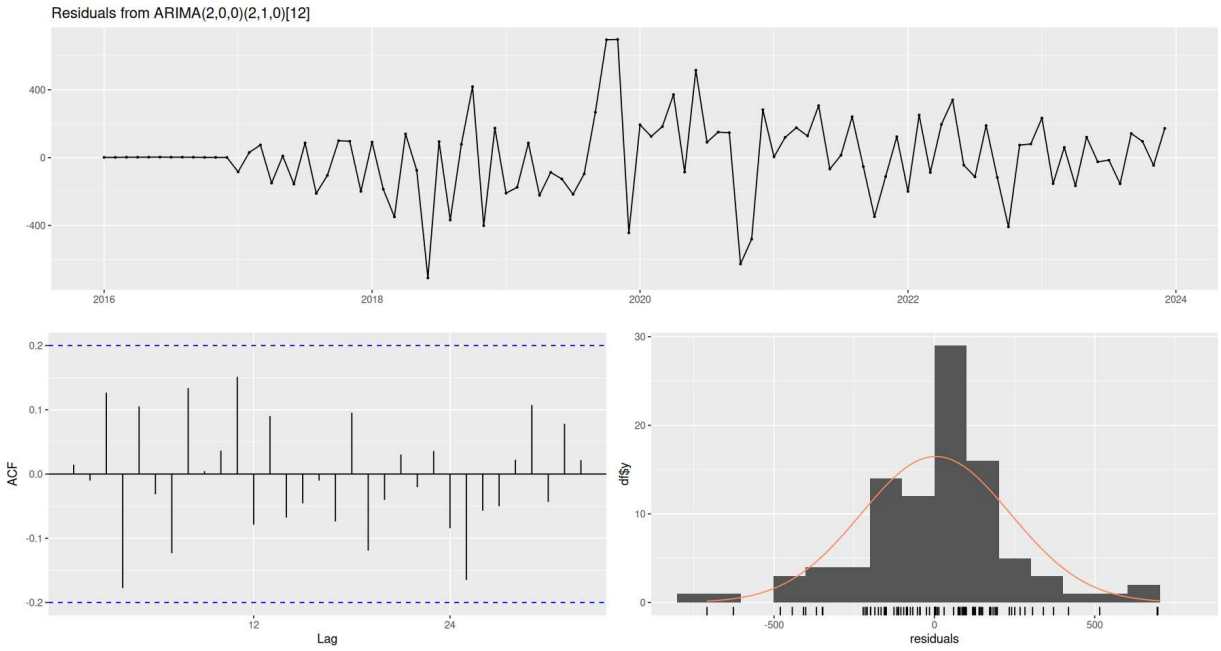
```r
# Forecast using SARIMA model
sarima_fc <- forecast(sarima_fit, h = 12)

# Plot the forecast
autoplot(sarima_fc)
```



5.3 Validación de residuos

In [ ]:
```r
checkresiduals(sarima_fit)
```

```
	Ljung-Box test

data:  Residuals from ARIMA(2,0,0)(2,1,0)[12]
Q* = 18.204, df = 15, p-value = 0.2521

Model df: 4.   Total lags used: 19
```



Los residuos del modelo SARIMA ajustado presentan las siguientes características:

- En la serie temporal de residuos, no se observa un patrón sistemático evidente. Sin embargo, hay algunos valores extremos (outliers), posiblemente vinculados al impacto del COVID-19.
- En el gráfico de ACF de los residuos, la mayoría de los rezagos se encuentran dentro del intervalo de confianza, lo que sugiere ausencia de autocorrelación significativa.
- El histograma de los residuos muestra una distribución ligeramente asimétrica, pero con forma aproximadamente normal.

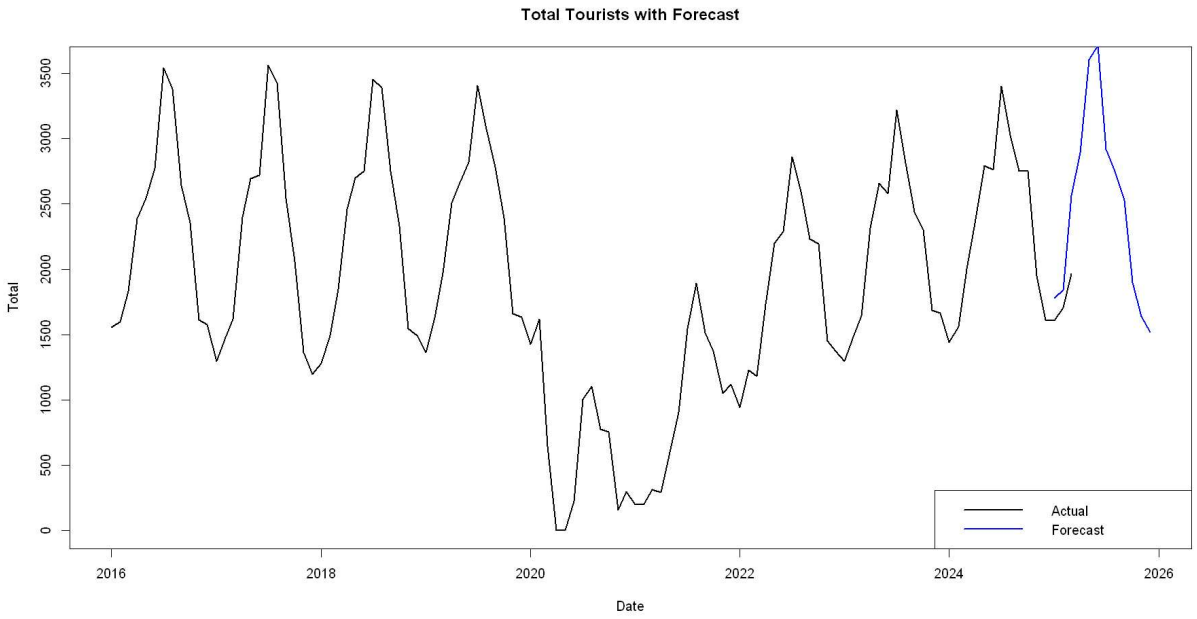
En conjunto, estos resultados indican que los residuos se comportan razonablemente como ruido blanco, por lo que el modelo no parece dejar estructura sin modelar.

5.4 Predicción y evaluación con SARIMA

In [128…

```
# Generate full 2025 dates
future_dates <- seq.Date(from = as.Date("2025-01-01"), by = "month", length.out = 1

# Create forecast frame with actuals where available
comparison <- data.frame(
  CleanDate = future_dates,
  Forecast = as.numeric(sarima_fc$mean)
) %>%
  left_join(df %>% select(CleanDate, Total), by = "CleanDate") %>%
  rename(Actual = Total)

# Add missing months to df
df_extended <- df %>%
  full_join(comparison, by = "CleanDate") %>%
  arrange(CleanDate)
```

```
# Plot
par(bg = "white")
plot(df_extended$CleanDate, df_extended$Total, type = "l", col = "black", lwd = 2,
lines(df_extended$CleanDate, df_extended$Forecast, col = "blue", lwd = 2)
legend("bottomright", legend = c("Actual", "Forecast"), col = c("black", "blue"), l
```



In [129…

```
comparison
```

A data.frame: 12 × 3

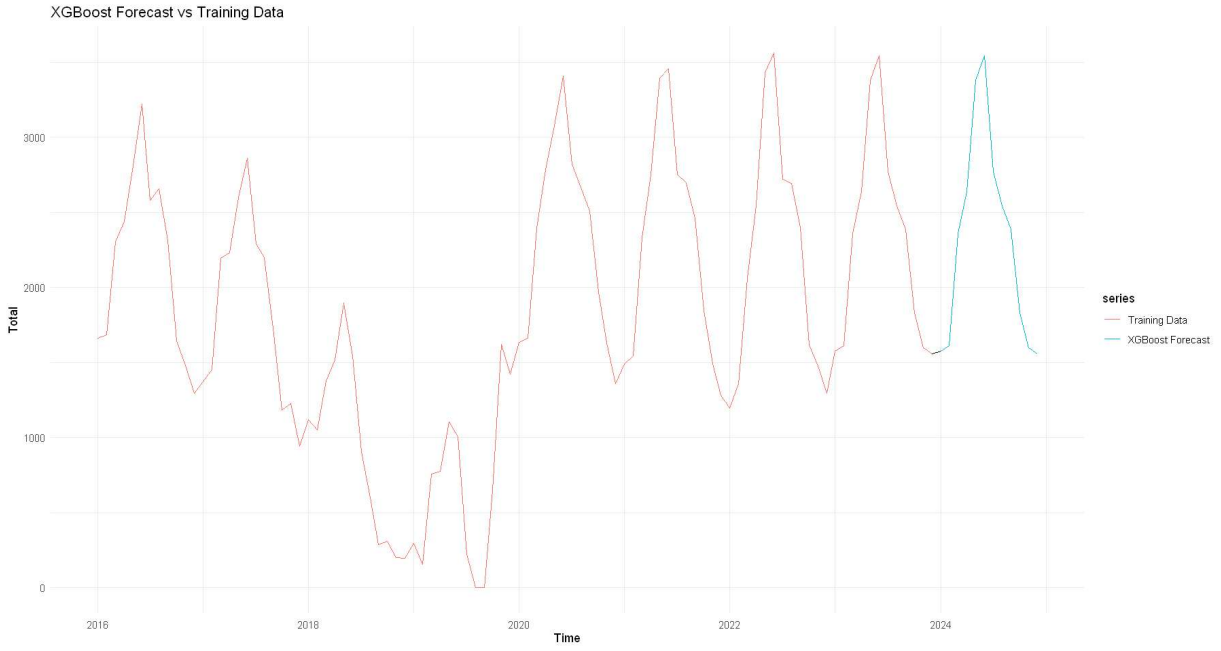
CleanDate	Forecast	Actual
<date>	<dbl>	<dbl>
2025-01-01	1776.899	1607.3
2025-02-01	1839.340	1704.5
2025-03-01	2564.050	1968.7
2025-04-01	2892.668	NA
2025-05-01	3607.926	NA
2025-06-01	3713.395	NA
2025-07-01	2920.858	NA
2025-08-01	2753.238	NA
2025-09-01	2531.048	NA
2025-10-01	1899.632	NA
2025-11-01	1640.387	NA
2025-12-01	1516.556	NA

6. XGBOOST

In [130…

```r
# Create a ts object of the XGBoost forecast
start_fc <- c(year(max(train_df$CleanDate)) + ifelse(month(max(train_df$CleanDate))
              (month(max(train_df$CleanDate)) %% 12) + 1)
xgb_ts <- ts(preds_xgb, start = start_fc, frequency = 12)

# Combine historical and forecast for plotting
combined <- ts(c(train_ts, preds_xgb), start = start(train_ts), frequency = 12)

# Plot
autoplot(combined) +
  autolayer(train_ts, series = "Training Data") +
  autolayer(xgb_ts, series = "XGBoost Forecast", PI = FALSE) +
  labs(title = "XGBoost Forecast vs Training Data",
       y = "Total", x = "Time") +
  theme_minimal()
```

```
Warning message in ggplot2::geom_line(ggplot2::aes(x = .data[["timeVal"]], y = .data
[["seriesVal"]], :
"Ignoring unknown parameters: `PI`"
```



7. Evaluación comparativa de modelos

7.1 Evaluación comparativa de modelos

In [131…

```r
library(Metrics)

# 1. Naive model
comparison_naive <- data.frame(
  CleanDate = future_dates,
```

```
  Forecast = as.numeric(naive_model$mean)
) %>%
  left_join(df %>% select(CleanDate, Total), by = "CleanDate") %>%
  rename(Actual = Total)

# 2. Holt-Winters
comparison_hw <- data.frame(
  CleanDate = future_dates,
  Forecast = as.numeric(hw_forecast$mean)
) %>%
  left_join(df %>% select(CleanDate, Total), by = "CleanDate") %>%
  rename(Actual = Total)

# 3. TSLM
comparison_tslm <- data.frame(
  CleanDate = future_dates,
  Forecast = as.numeric(fc$mean)
) %>%
  left_join(df %>% select(CleanDate, Total), by = "CleanDate") %>%
  rename(Actual = Total)

# 4. XGBoost
comparison_xgboost <- data.frame(
  CleanDate = future_dates,
  Forecast = preds_xgb
) %>%
  left_join(df %>% select(CleanDate, Total), by = "CleanDate") %>%
  rename(Actual = Total)
```

In [133…

```
epam <- function(actual, forecast) {
  mean(abs((actual - forecast) / actual), na.rm = TRUE) * 100
}

get_errors <- function(df) {
  df_clean <- df %>% filter(!is.na(Actual), !is.na(Forecast), Actual != 0)
  c(
    MAE = round(mae(df_clean$Actual, df_clean$Forecast), 2),
    RMSE = round(rmse(df_clean$Actual, df_clean$Forecast), 2),
    EPAM = round(epam(df_clean$Actual, df_clean$Forecast), 2)
  )
}
# Obtener métricas
metrics_naive <- get_errors(comparison_naive)
metrics_hw <- get_errors(comparison_hw)
metrics_tslm <- get_errors(comparison_tslm)
metrics_sarima <- get_errors(comparison)
metrics_xgboost <- get_errors(comparison_xgboost)
```

In [135…

```
# Construir tabla
model_comparison <- data.frame(
  Modelo = c("Naive", "Holt-Winters", "TSLM", "SARIMA", "XGBoost"),
  MAE = c(metrics_naive["MAE"], metrics_hw["MAE"], metrics_tslm["MAE"], metrics_sar
  RMSE = c(metrics_naive["RMSE"], metrics_hw["RMSE"], metrics_tslm["RMSE"], metrics
  EPAM = c(metrics_naive["EPAM"], metrics_hw["EPAM"], metrics_tslm["EPAM"], metrics
)
```

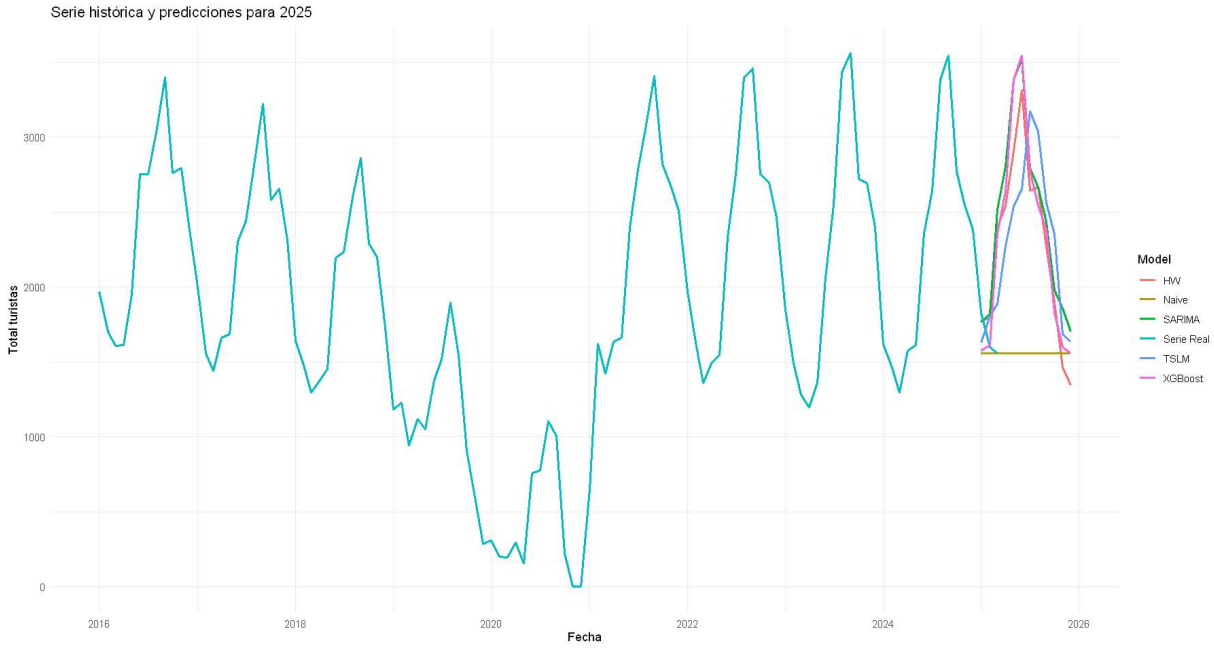
```
print(model_comparison)
```

```
       Modelo    MAE   RMSE  EPAM
1       Naive 203.97 254.79 10.94
2 Holt-Winters 229.78 272.76 12.54
3        TSLM  61.85  67.46  3.49
4      SARIMA 299.93 365.78 16.23
5     XGBoost 169.91 231.21  8.98
```

7.2 Predicción final para el año 2025

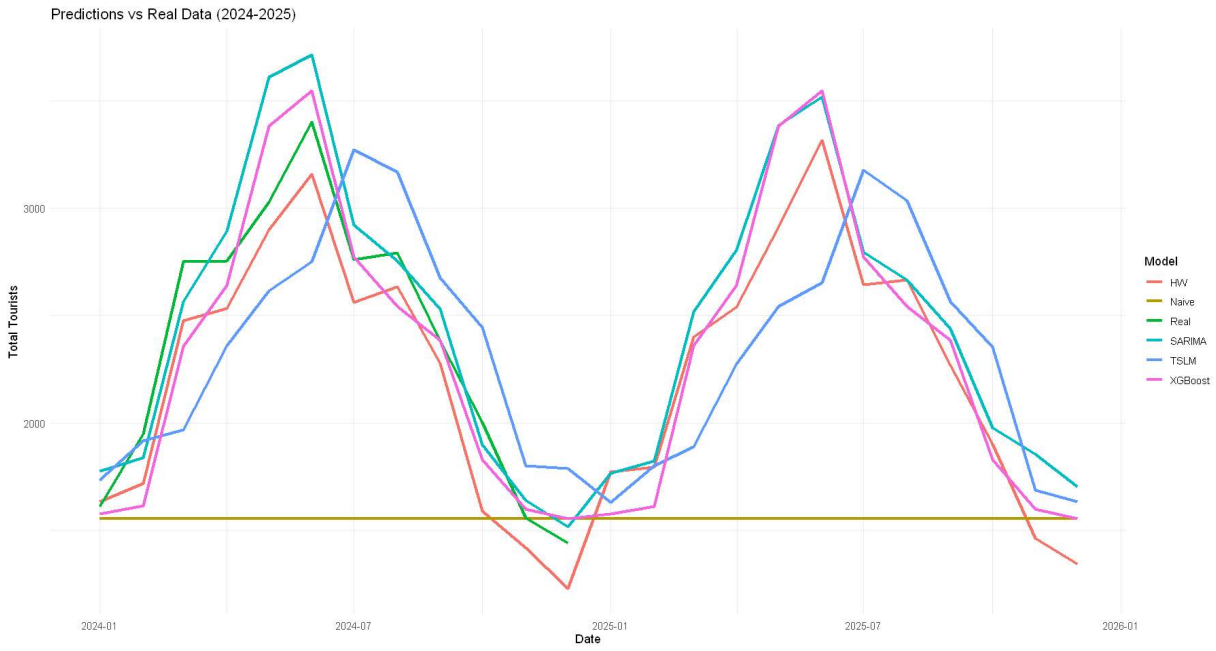
In [ ]:
```
# Entrenar con TODOS los datos disponibles
full_train_df <- df %>%
  filter(year(CleanDate) < 2025)
full_train_ts <- ts(full_train_df$Total,
                    start = c(year(min(full_train_df$CleanDate)), month(min(full_tr
                    frequency = 12)

# Holt-Winters
hw_model_2025 <- HoltWinters(full_train_ts)
hw_fc_2025 <- forecast(hw_model_2025, h = 12)

# Naive
naive_model_2025 <- naive(full_train_ts, h = 12)

# TSLM
train_df_tslm <- full_train_df %>%
  mutate(Date = as.Date(CleanDate),
         covid_dummy = if_else(year(Date) == 2020 | (year(Date) == 2021 & month(Dat
         trend = row_number(),
         season = factor(month(Date), levels = 1:12))

train_ts_tslm <- ts(train_df_tslm$Total,
                    start = c(year(min(train_df_tslm$Date)), month(min(train_df_tsl
                    frequency = 12)

fit_tslm <- tslm(train_ts_tslm ~ trend + season + covid_dummy, data = train_df_tslm

newdata_2025 <- data.frame(
  trend = nrow(train_df_tslm) + 1:12,
  season = factor(1:12, levels = 1:12),
  covid_dummy = 0
)
tslm_fc_2025 <- forecast(fit_tslm, newdata = newdata_2025)

# SARIMA
sarima_model_2025 <- auto.arima(full_train_ts)
sarima_fc_2025 <- forecast(sarima_model_2025, h = 12)

# XGBoost
lags <- 12
vec <- as.numeric(full_train_ts)
df_lagged <- data.frame(y = vec)
for (i in 1:lags) {
```

```r
  df_lagged[[paste0("lag_", i)]] <- stats::lag(vec, -i)
}
df_lagged <- na.omit(df_lagged)

dtrain <- xgb.DMatrix(data = as.matrix(df_lagged[, -1]), label = df_lagged$y)
xgb_model <- xgboost(data = dtrain, nrounds = 100, objective = "reg:squarederror",

x_input <- tail(vec, lags)
preds_xgb_2025 <- c()
for (i in 1:12) {
  dtest <- xgb.DMatrix(matrix(x_input[(length(x_input) - lags + 1):length(x_input)]
  pred <- predict(xgb_model, dtest)
  preds_xgb_2025 <- c(preds_xgb_2025, pred)
  x_input <- c(x_input, pred)
}

# Crear DataFrame de resultados
forecast_dates_2025 <- seq(as.Date("2025-01-01"), by = "month", length.out = 12)
pred_df_2025 <- data.frame(
  Date = forecast_dates_2025,
  SARIMA = as.numeric(sarima_fc_2025$mean),
  HW = as.numeric(hw_fc_2025$mean),
  TSLM = as.numeric(tslm_fc_2025$mean),
  Naive = as.numeric(naive_model_2025$mean),
  XGBoost = preds_xgb_2025
)

# Crear serie original como data.frame
original_df <- data.frame(
  Date = seq(min(df$CleanDate), by = "month", length.out = length(ts_total)),
  Total = as.numeric(ts_total)
)

# Convertir predicciones a formato largo y renombrar columnas
df_long_2025 <- pred_df_2025 %>%
  pivot_longer(cols = -Date, names_to = "Model", values_to = "Forecast")

# Añadir serie original como "Real"
original_long <- original_df %>%
  rename(Forecast = Total) %>%
  mutate(Model = "Serie Real")

# Combinar original + predicciones
combined_plot_df <- bind_rows(original_long, df_long_2025)

# Graficar
ggplot(combined_plot_df, aes(x = Date, y = Forecast, color = Model)) +
  geom_line(size = 1.1) +
  labs(title = "Serie histórica y predicciones para 2025",
       y = "Total turistas", x = "Fecha") +
  theme_minimal()
```



7.3 Visualización comparativa 2024–2025

In [ ]:
```
# Filter data for 2024 and 2025
comparison_df_full <- comparison_df %>%
    filter(Date >= as.Date("2024-01-01") & Date <= as.Date("2025-12-31"))

# Add predictions for 2025
pred_df_2025_long <- pred_df_2025 %>%
    pivot_longer(cols = -Date, names_to = "Model", values_to = "Forecast")

comparison_df_full_long <- comparison_df_full %>%
    pivot_longer(cols = -c(Date, Real), names_to = "Model", values_to = "Forecast") %>%
    bind_rows(pred_df_2025_long)

# Combine real data for 2024 and 2025
real_data <- comparison_df_full %>%
    select(Date, Real) %>%
    rename(Forecast = Real) %>%
    mutate(Model = "Real")

# Combine all data for plotting
plot_data <- bind_rows(real_data, comparison_df_full_long)

# Plot
ggplot(plot_data, aes(x = Date, y = Forecast, color = Model)) +
    geom_line(size = 1.2) +
    labs(title = "Predictions vs Real Data (2024-2025)",
             x = "Date", y = "Total Tourists") +
    theme_minimal()
```



8. Discusión y conclusiones

De acuerdo con estas métricas:

- El modelo determinista TSLM fue el que obtuvo los errores más bajos tanto en MAE como en RMSE, lo que indica una mayor precisión predictiva.
- El modelo Naive, aunque simple, se comportó mejor que Holt-Winters y SARIMA, mostrando que la serie es muy predecible por arrastre estacional.
- El modelo SARIMA, a pesar de ser más complejo y haber pasado la validación de residuos, presentó el peor rendimiento en predicción real (2025).

Por tanto, se concluye que el modelo TSLM con tendencia, estacionalidad mensual y efecto COVID es el más adecuado para predecir esta serie. Además, su interpretación es directa, y su comportamiento frente a eventos estructurales (como la pandemia) puede ajustarse fácilmente mediante dummies.

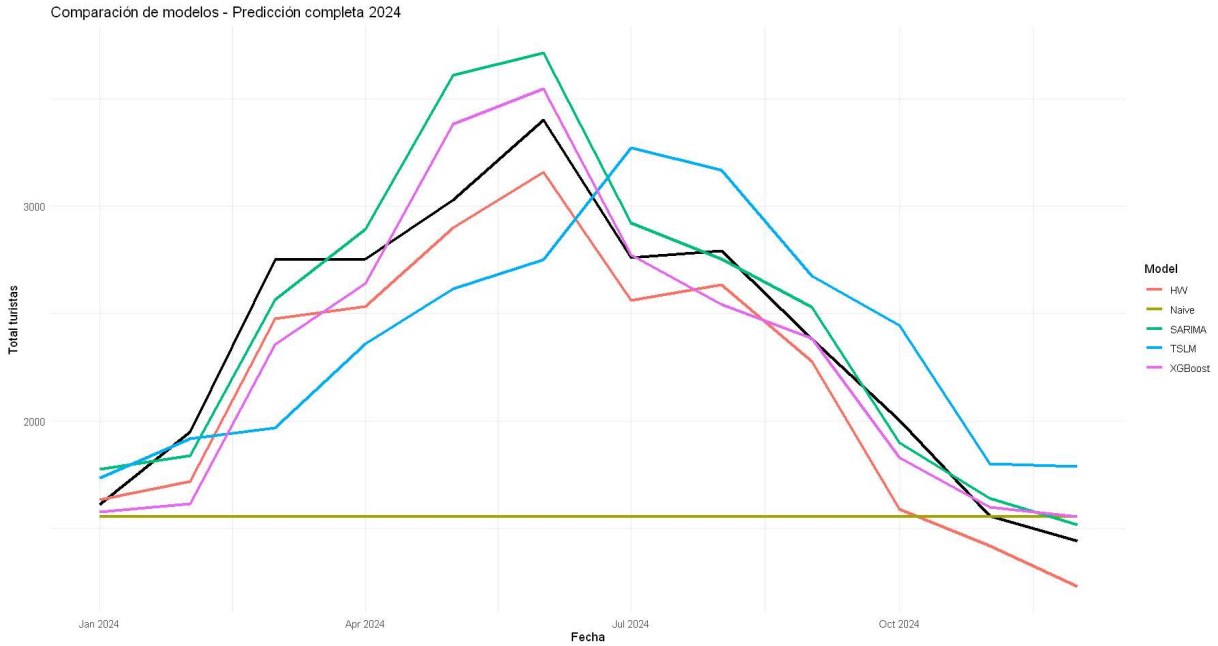
9. Extras

In [ ]:
```
# Evaluación extramuestral sobre el año 2024

# Crear conjuntos de entrenamiento y prueba
train_df <- df %>% filter(year(CleanDate) < 2024)
test_df <- df %>% filter(year(CleanDate) == 2024)

# Crear serie temporal
train_ts <- ts(train_df$Total, start = c(year(min(train_df$CleanDate)), month(min(t
```

```r
# Holt-Winters
hw_model <- HoltWinters(train_ts)
hw_fc <- forecast(hw_model, h = 12)

# Naive
naive_model <- naive(train_ts, h = 12)

# TSLM
train_df_tslm <- train_df %>%
  mutate(Date = as.Date(CleanDate),
         covid_dummy = if_else(year(Date) == 2020 | (year(Date) == 2021 & month(Dat
         trend = row_number(),
         season = factor(month(Date), levels = 1:12))

train_ts_tslm <- ts(train_df_tslm$Total, start = c(year(min(train_df_tslm$Date)), m
fit_tslm <- tslm(train_ts_tslm ~ trend + season + covid_dummy, data = train_df_tslm

newdata <- data.frame(
  trend = nrow(train_df_tslm) + 1:12,
  season = factor(1:12, levels = 1:12),
  covid_dummy = 0
)
tslm_fc <- forecast(fit_tslm, newdata = newdata)

# SARIMA (auto.arima)
sarima_model <- auto.arima(train_ts)
sarima_fc <- forecast(sarima_model, h = 12)

# XGBoost
lags <- 12
vec <- as.numeric(train_ts)
df_lagged <- data.frame(y = vec)
for (i in 1:lags) {
  df_lagged[[paste0("lag_", i)]] <- stats::lag(vec, -i)
}
df_lagged <- na.omit(df_lagged)

dtrain <- xgb.DMatrix(data = as.matrix(df_lagged[, -1]), label = df_lagged$y)
xgb_model <- xgboost(data = dtrain, nrounds = 100, objective = "reg:squarederror",

x_input <- tail(vec, lags)
preds_xgb <- c()
for (i in 1:12) {
  dtest <- xgb.DMatrix(matrix(x_input[(length(x_input) - lags + 1):length(x_input)]
  pred <- predict(xgb_model, dtest)
  preds_xgb <- c(preds_xgb, pred)
  x_input <- c(x_input, pred)
}

# Combinar resultados
forecast_dates <- seq(as.Date("2024-01-01"), by = "month", length.out = 12)
comparison_df <- data.frame(
  Date = forecast_dates,
  Real = test_df$Total,
  SARIMA = as.numeric(sarima_fc$mean),
  HW = as.numeric(hw_fc$mean),
```

```
  TSLM = as.numeric(tslm_fc$mean),
  Naive = as.numeric(naive_model$mean),
  XGBoost = preds_xgb
)

# Graficar
library(tidyr)
library(ggplot2)

df_long <- comparison_df %>%
  pivot_longer(cols = -c(Date, Real), names_to = "Model", values_to = "Forecast")

ggplot(df_long, aes(x = Date)) +
  geom_line(aes(y = Real), color = "black", size = 1.2, linetype = "solid") +
  geom_line(aes(y = Forecast, color = Model), size = 1.2) +
  labs(title = "Comparación de modelos - Predicción completa 2024", y = "Total turi
  theme_minimal()

# Calcular métricas
library(Metrics)
cat("MAE y RMSE para el año 2024:\n")
for (m in c("SARIMA", "HW", "TSLM", "Naive", "XGBoost")) {
  mae_val <- mae(comparison_df$Real, comparison_df[[m]])
  rmse_val <- rmse(comparison_df$Real, comparison_df[[m]])
  cat(m, "- MAE:", round(mae_val, 2), "| RMSE:", round(rmse_val, 2), "\n")
}
```

```
MAE y RMSE para el año 2024:
SARIMA - MAE: 175.28 | RMSE: 223.76
HW - MAE: 195.78 | RMSE: 216.86
TSLM - MAE: 383.95 | RMSE: 432.3
Naive - MAE: 832.41 | RMSE: 1019.15
XGBoost - MAE: 163.74 | RMSE: 211.18
```

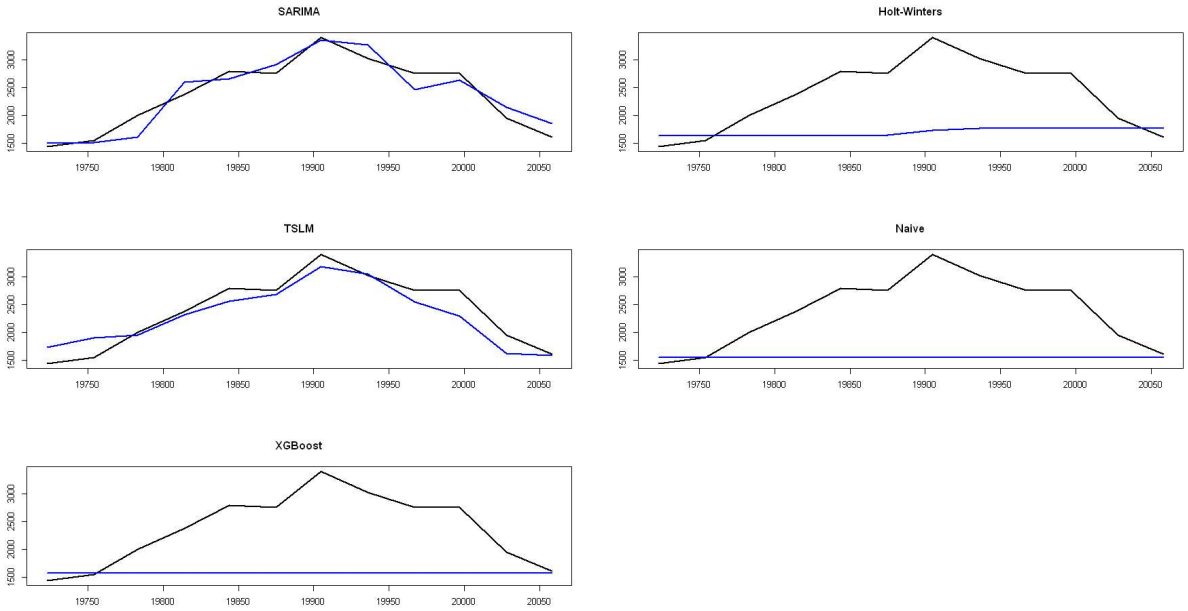


In [ ]:
```
# Validación tipo rolling forecast

# Inicializar resultados
```

```
rolling_hw <- data.frame()
rolling_tslm <- data.frame()
rolling_naive <- data.frame()
rolling_xgb <- data.frame()  # <- Añadido para XGBoost

for (current_date in seq(start_date, end_date, by = "month")) {
  train_df <- df %>% filter(CleanDate < current_date)
  test_df <- df %>% filter(CleanDate == current_date)
  if (nrow(test_df) == 0) next

  train_ts <- ts(train_df$Total, start = c(year(min(train_df$CleanDate)), month(min

  # Holt-Winters
  hw_model <- HoltWinters(train_ts)
  hw_forecast <- forecast(hw_model, h = 1)
  rolling_hw <- rbind(rolling_hw, data.frame(CleanDate = current_date, Forecast = a

  # TSLM
  train_df_tslm <- train_df %>%
    mutate(Date = as.Date(CleanDate),
           covid_dummy = if_else(year(Date) == 2020 | (year(Date) == 2021 & month(D
           trend = row_number(),
           season = factor(month(Date), levels = 1:12))

  train_ts_tslm <- ts(train_df_tslm$Total, start = c(year(min(train_df_tslm$Date)),
  fit_tslm <- tslm(train_ts_tslm ~ trend + season + covid_dummy, data = train_df_ts
  newdata <- data.frame(trend = nrow(train_df_tslm) + 1,
                        season = factor(month(as.Date(current_date)), levels = leve
                        covid_dummy = 0)
  tslm_fc <- forecast(fit_tslm, newdata = newdata)
  rolling_tslm <- rbind(rolling_tslm, data.frame(CleanDate = current_date, Forecast

  # Naive
  naive_model <- naive(train_ts, h = 1)
  rolling_naive <- rbind(rolling_naive, data.frame(CleanDate = current_date, Foreca

  # XGBoost
  lags <- 12
  vec <- as.numeric(train_ts)
  if (length(vec) <= lags) next  # no se puede aplicar XGBoost si hay pocos datos

  df_lagged <- data.frame(y = vec)
  for (i in 1:lags) {
    df_lagged[[paste0("lag_", i)]] <- stats::lag(vec, -i)
  }
  df_lagged <- na.omit(df_lagged)

  dtrain <- xgb.DMatrix(data = as.matrix(df_lagged[, -1]), label = df_lagged$y)
  xgb_model <- xgboost(data = dtrain, nrounds = 100, objective = "reg:squarederror"

  # Construir observación actual con rezagos
  x_input <- tail(vec, lags)
  dtest <- xgb.DMatrix(matrix(x_input, nrow = 1))
  pred_xgb <- predict(xgb_model, dtest)

  rolling_xgb <- rbind(rolling_xgb, data.frame(CleanDate = current_date, Forecast =
```

```
}

# Plot comparisons
par(mfrow = c(3, 2), bg = "white")
plot(rolling_predictions$CleanDate, rolling_predictions$Actual, type = "l", col = "
lines(rolling_predictions$CleanDate, rolling_predictions$Forecast, col = "blue", lw

plot(rolling_hw$CleanDate, rolling_hw$Actual, type = "l", col = "black", lwd = 2, m
lines(rolling_hw$CleanDate, rolling_hw$Forecast, col = "blue", lwd = 2)

plot(rolling_tslm$CleanDate, rolling_tslm$Actual, type = "l", col = "black", lwd =
lines(rolling_tslm$CleanDate, rolling_tslm$Forecast, col = "blue", lwd = 2)

plot(rolling_naive$CleanDate, rolling_naive$Actual, type = "l", col = "black", lwd
lines(rolling_naive$CleanDate, rolling_naive$Forecast, col = "blue", lwd = 2)

plot(rolling_xgb$CleanDate, rolling_xgb$Actual, type = "l", col = "black", lwd = 2,
lines(rolling_xgb$CleanDate, rolling_xgb$Forecast, col = "blue", lwd = 2)

# Compute and print error metrics
cat("Model Performance for 2024:\n")
cat("SARIMA MAE:", mae(rolling_predictions$Actual, rolling_predictions$Forecast), "
cat("Holt-Winters MAE:", mae(rolling_hw$Actual, rolling_hw$Forecast), "RMSE:", rmse
cat("TSLM MAE:", mae(rolling_tslm$Actual, rolling_tslm$Forecast), "RMSE:", rmse(rol
cat("Naive MAE:", mae(rolling_naive$Actual, rolling_naive$Forecast), "RMSE:", rmse(
cat("XGBoost MAE:", mae(rolling_xgb$Actual, rolling_xgb$Forecast), "RMSE:", rmse(ro
```

```
Model Performance for 2024:
SARIMA MAE: 178.4726 RMSE: 205.3457
Holt-Winters MAE: 737.3344 RMSE: 894.3202
TSLM MAE: 193.2547 RMSE: 238.0035
Naive MAE: 832.4083 RMSE: 1019.149
XGBoost MAE: 818.2532 RMSE: 1002.634
```



7.5