ECON 144 HW 4

Leonard Zhu

2024-11-21

```
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(ggplot2)
library(vars)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: strucchange

## Warning: package 'strucchange' was built under R version 4.4.1

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

```
## Loading required package: sandwich

## Warning: package 'sandwich' was built under R version 4.4.1

## Loading required package: urca

## Loading required package: lmtest
```

```
library(lmtest)
```

```
# Load data
data <- read_xlsx("~/Downloads/msa1_data.xlsx")
msa1_data <- read_xlsx("~/Downloads/msa1_data.xlsx")
msa2_data <- read_xlsx("~/Downloads/msa2_data.xlsx")

# Ensure the data is sorted by year and month
msa1_data <- msa1_data %>% arrange(Year, Month)
msa2_data <- msa2_data %>% arrange(Year, Month)

# Calculate price growth for both MSAs (log difference of the index)
msa1_data <- msa1_data %>% mutate(growth = c(NA, diff(log(Index_SA))))
msa2_data <- msa2_data %>% mutate(growth = c(NA, diff(log(Index_SA))))

# Combine both datasets for analysis
combined_data <- data.frame(
  Year = msa1_data$Year,
  Month = msa1_data$Month,
  MSA1_growth = msa1_data$growth,
  MSA2_growth = msa2_data$growth
)

# Remove rows with NA values
combined_data <- combined_data %>% drop_na()

# Split data into estimation (training) and prediction (test) samples
set.seed(123)
n <- nrow(combined_data)
test_size <- 20  # At least 20 observations for the prediction sample
train_data <- combined_data[1:(n - test_size), ]
test_data <- combined_data[(n - test_size + 1):n, ]
```

```
# Select optimal lag length
lag_selection <- VARselect(train_data[, c("MSA1_growth", "MSA2_growth")], lag.max = 10, type = "const")
optimal_lags <- lag_selection$selection["AIC(n)"]  # Using AIC criterion

# Fit the VAR model
var_model <- VAR(train_data[, c("MSA1_growth", "MSA2_growth")], p = optimal_lags, type = "const")

# Summary of the VAR model
summary(var_model)
```

```
##
```

```
## VAR Estimation Results:
## ========================
## Endogenous variables: MSA1_growth, MSA2_growth
## Deterministic variables: const
## Sample size: 566
## Log Likelihood: 5393.805
## Roots of the characteristic polynomial:
## 0.9455 0.9186 0.9186 0.9104 0.9104 0.9049 0.9049 0.8952 0.8952 0.8516 0.8516 0.8392 0.8392 0.7803 0.7
## Call:
## VAR(y = train_data[, c("MSA1_growth", "MSA2_growth")], p = optimal_lags,
##     type = "const")
##
##
## Estimation results for equation MSA1_growth:
## ============================================
## MSA1_growth = MSA1_growth.l1 + MSA2_growth.l1 + MSA1_growth.l2 + MSA2_growth.l2 + MSA1_growth.l3 + MS
##
##                    Estimate Std. Error t value Pr(>|t|)
## MSA1_growth.l1   0.1205528  0.2562476   0.470 0.638219
## MSA2_growth.l1   1.1570331  0.2519202   4.593 5.44e-06 ***
## MSA1_growth.l2   0.4591980  0.3718749   1.235 0.217430
## MSA2_growth.l2  -1.7560850  0.3728243  -4.710 3.14e-06 ***
## MSA1_growth.l3   0.5078513  0.3702732   1.372 0.170765
## MSA2_growth.l3   0.8253475  0.3733149   2.211 0.027460 *
## MSA1_growth.l4   0.0569424  0.3637223   0.157 0.875654
## MSA2_growth.l4  -1.0486251  0.3675333  -2.853 0.004493 **
## MSA1_growth.l5  -1.4671030  0.3947702  -3.716 0.000223 ***
## MSA2_growth.l5   2.1826434  0.4106152   5.316 1.55e-07 ***
## MSA1_growth.l6   0.5401907  0.3984772   1.356 0.175776
## MSA2_growth.l6  -0.7030415  0.4172318  -1.685 0.092558 .
## MSA1_growth.l7   0.1267132  0.3608285   0.351 0.725594
## MSA2_growth.l7  -0.1012620  0.3674935  -0.276 0.783000
## MSA1_growth.l8   1.0866003  0.3601910   3.017 0.002674 **
## MSA2_growth.l8  -1.0368593  0.3697426  -2.804 0.005223 **
## MSA1_growth.l9  -0.8130673  0.3679103  -2.210 0.027523 *
## MSA2_growth.l9   0.7003014  0.3746141   1.869 0.062104 .
## MSA1_growth.l10  0.3259443  0.2533544   1.287 0.198810
## MSA2_growth.l10 -0.2563370  0.2496386  -1.027 0.304955
## const            0.0004612  0.0002593   1.779 0.075813 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.00522 on 545 degrees of freedom
## Multiple R-Squared: 0.8166,  Adjusted R-squared: 0.8098
## F-statistic: 121.3 on 20 and 545 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation MSA2_growth:
## ============================================
## MSA2_growth = MSA1_growth.l1 + MSA2_growth.l1 + MSA1_growth.l2 + MSA2_growth.l2 + MSA1_growth.l3 + MS
##
##                    Estimate Std. Error t value Pr(>|t|)
## MSA1_growth.l1  -0.9807041  0.2601915  -3.769 0.000182 ***
```

```
## MSA2_growth.l1   2.2710062  0.2557975   8.878  < 2e-16 ***
## MSA1_growth.l2   0.2949964  0.3775983   0.781 0.434998
## MSA2_growth.l2  -1.6756975  0.3785623  -4.426 1.16e-05 ***
## MSA1_growth.l3   0.4344566  0.3759720   1.156 0.248369
## MSA2_growth.l3   0.9659124  0.3790605   2.548 0.011102 *
## MSA1_growth.l4   0.6497220  0.3693202   1.759 0.079098 .
## MSA2_growth.l4  -1.7324720  0.3731899  -4.642 4.32e-06 ***
## MSA1_growth.l5  -1.5860114  0.4008460  -3.957 8.61e-05 ***
## MSA2_growth.l5   2.4165067  0.4169349   5.796 1.15e-08 ***
## MSA1_growth.l6   0.5547863  0.4046101   1.371 0.170889
## MSA2_growth.l6  -0.7965475  0.4236533  -1.880 0.060616 .
## MSA1_growth.l7  -0.0052206  0.3663820  -0.014 0.988636
## MSA2_growth.l7   0.1115461  0.3731495   0.299 0.765106
## MSA1_growth.l8   1.3381840  0.3657346   3.659 0.000278 ***
## MSA2_growth.l8  -1.3333699  0.3754333  -3.552 0.000416 ***
## MSA1_growth.l9  -1.2450397  0.3735728  -3.333 0.000918 ***
## MSA2_growth.l9   1.1595898  0.3803797   3.049 0.002411 **
## MSA1_growth.l10  0.6298852  0.2572538   2.448 0.014659 *
## MSA2_growth.l10 -0.5696044  0.2534807  -2.247 0.025031 *
## const            0.0005075  0.0002633   1.928 0.054402 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.0053 on 545 degrees of freedom
## Multiple R-Squared: 0.8377,  Adjusted R-squared: 0.8318
## F-statistic: 140.7 on 20 and 545 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##             MSA1_growth MSA2_growth
## MSA1_growth   2.725e-05   2.731e-05
## MSA2_growth   2.731e-05   2.809e-05
##
## Correlation matrix of residuals:
##             MSA1_growth MSA2_growth
## MSA1_growth      1.0000      0.9872
## MSA2_growth      0.9872      1.0000
```

```
# Test if MSA1 growth Granger-causes MSA2 growth
granger_test1 <- causality(var_model, cause = "MSA1_growth")
granger_test1
```

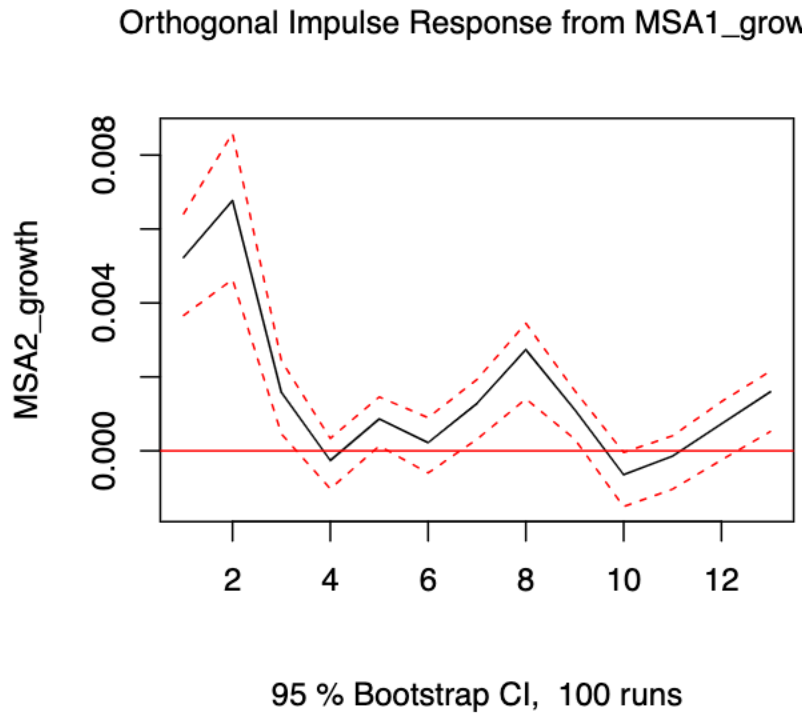
```
## $Granger
##
##  Granger causality H0: MSA1_growth do not Granger-cause MSA2_growth
##
## data:  VAR object var_model
## F-Test = 11.457, df1 = 10, df2 = 1090, p-value < 2.2e-16
##
##
## $Instant
##
```

```
##  H0: No instantaneous causality between: MSA1_growth and MSA2_growth
##
## data:  VAR object var_model
## Chi-squared = 279.35, df = 1, p-value < 2.2e-16
```

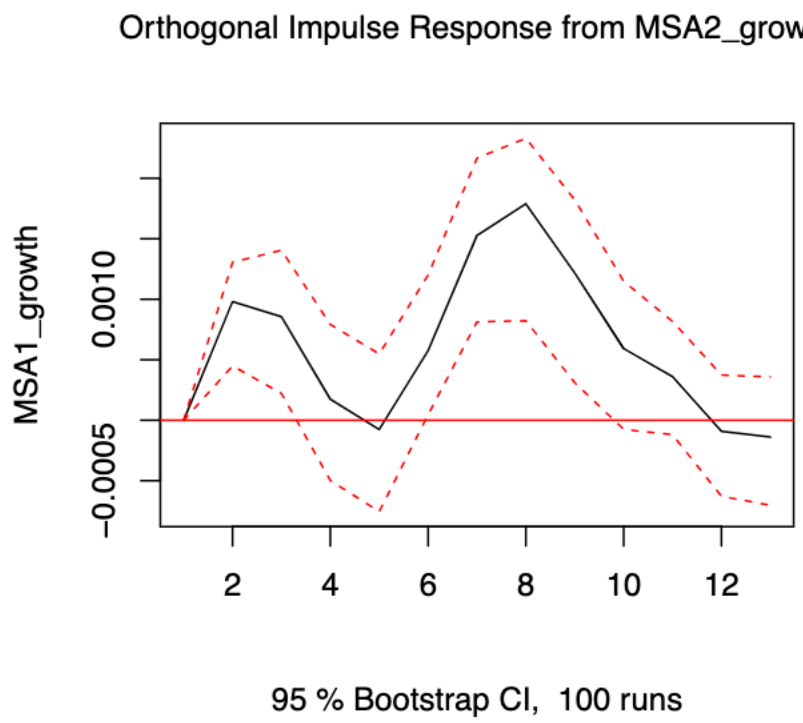
```
# Test if MSA2 growth Granger-causes MSA1 growth
granger_test2 <- causality(var_model, cause = "MSA2_growth")
granger_test2
```

```
## $Granger
##
##  Granger causality H0: MSA2_growth do not Granger-cause MSA1_growth
##
## data:  VAR object var_model
## F-Test = 9.0008, df1 = 10, df2 = 1090, p-value = 2.309e-14
##
##
## $Instant
##
##  H0: No instantaneous causality between: MSA2_growth and MSA1_growth
##
## data:  VAR object var_model
## Chi-squared = 279.35, df = 1, p-value < 2.2e-16
```

```
# Compute and plot impulse-response functions
irf <- irf(var_model, impulse = "MSA1_growth", response = "MSA2_growth", n.ahead = 12, boot = TRUE)
plot(irf)
```




```
# Repeat for other combinations
irf2 <- irf(var_model, impulse = "MSA2_growth", response = "MSA1_growth", n.ahead = 12, boot = TRUE)
plot(irf2)
```




```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## v purrr     1.0.2     v tibble    3.2.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x stringr::boundary() masks strucchange::boundary()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x MASS::select()      masks dplyr::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(forecast)
```

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

```
library(tsibble)
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
```

```
##
## Attaching package: 'tsibble'
##
## The following object is masked from 'package:lubridate':
##
##     interval
##
## The following object is masked from 'package:zoo':
##
##     index
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, union
```

```
library(feasts)
```

```
## Warning: package 'feasts' was built under R version 4.4.1
```

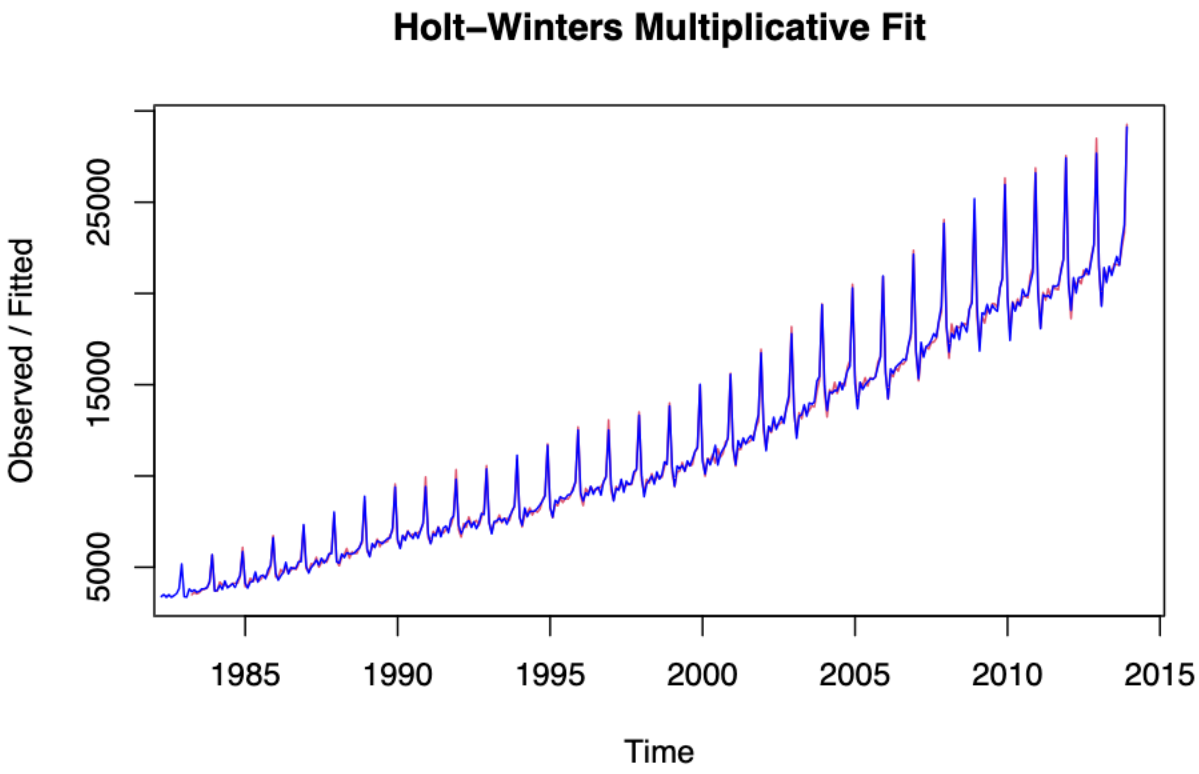
```
## Loading required package: fabletools
```

```
## Warning: package 'fabletools' was built under R version 4.4.1
```

```
library(stats)

# Load the data
library(readxl)
retail_data <- read_excel("~/Downloads/retail.xlsx", sheet = "Data1")

# Convert to a time series object
turnover_ts <- ts(retail_data$Turnover, start = c(1982, 4), frequency = 12)

# Display the first few rows of the data
head(turnover_ts)
```

```
##         Apr    May    Jun    Jul    Aug    Sep
## 1982 3396.4 3497.9 3357.8 3486.8 3355.9 3454.3
```

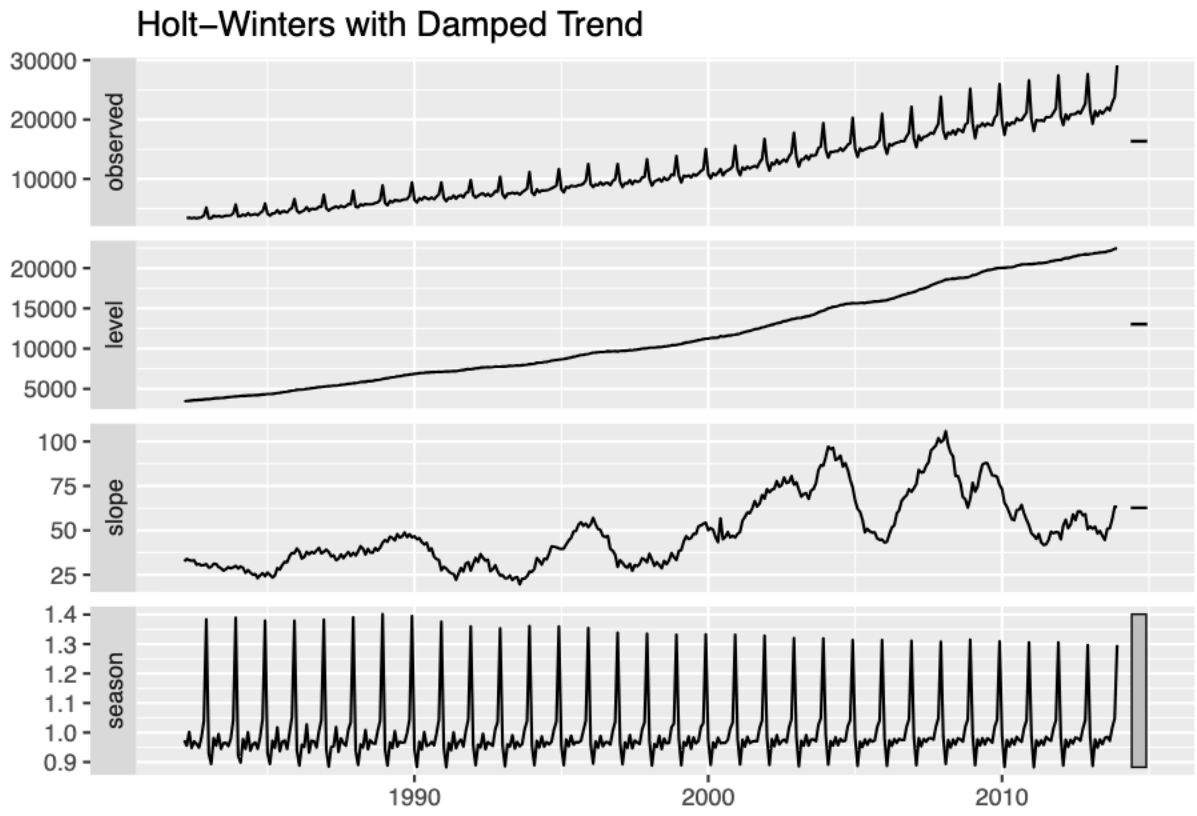
```
# Apply Holt-Winters' multiplicative method
hw_model <- HoltWinters(turnover_ts, seasonal = "multiplicative")

# Plot the fitted model using base R
plot(hw_model, main = "Holt-Winters Multiplicative Fit", col = "blue", lty = 1)
```




```
# Summarize the model
summary(hw_model)
```

```
##              Length Class  Mode
## fitted       1476   mts    numeric
## x             381   ts     numeric
## alpha           1   -none- numeric
## beta            1   -none- numeric
## gamma           1   -none- numeric
## coefficients   14   -none- numeric
## seasonal        1   -none- character
## SSE             1   -none- numeric
## call            3   -none- call
```

```
# Fit the damped Holt-Winters model
hw_damped <- ets(turnover_ts, model = "MAM")

# Plot the damped trend model
autoplot(hw_damped) +
  ggtitle("Holt-Winters with Damped Trend")
```




```
# Calculate one-step forecasts
fitted_hw <- fitted(hw_model)
fitted_damped <- fitted(hw_damped)

# Compute RMSE
rmse_hw <- sqrt(mean((turnover_ts - fitted_hw[,1])^2, na.rm = TRUE))
rmse_damped <- sqrt(mean((turnover_ts - fitted_damped)^2, na.rm = TRUE))

# Display RMSE
rmse_hw
```

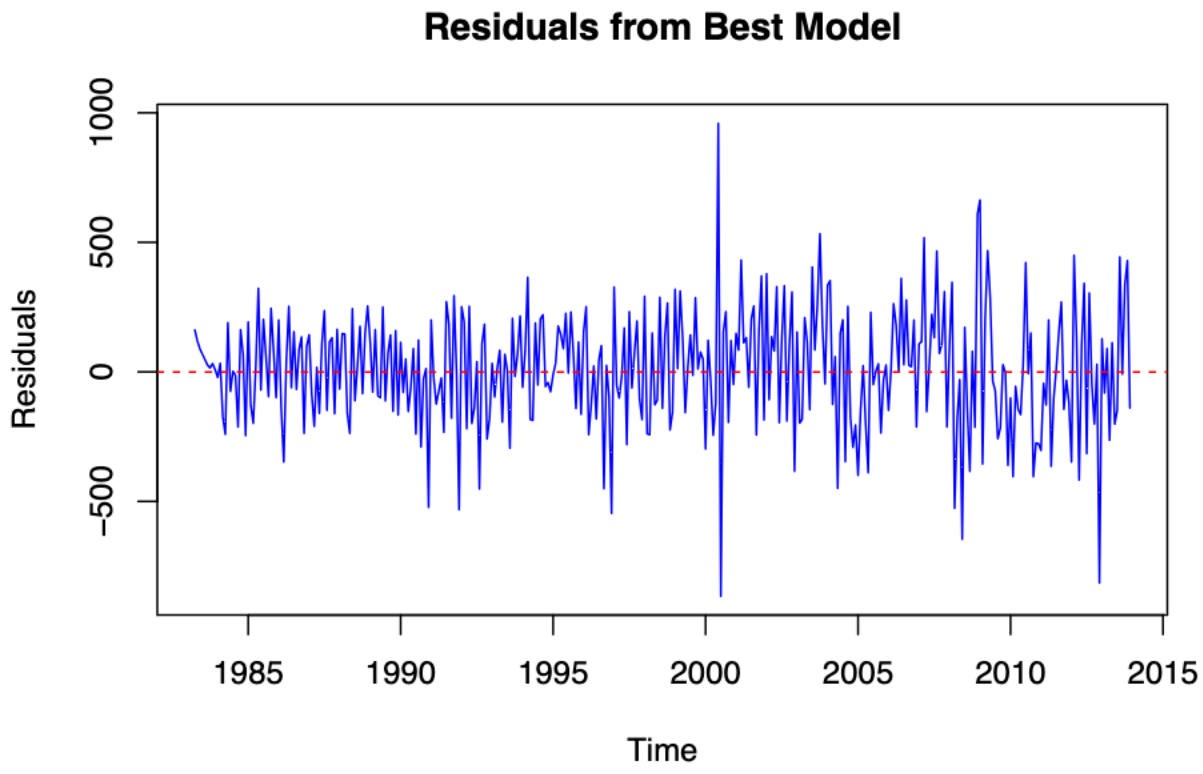
```
## [1] 227.0205
```

```
rmse_damped
```

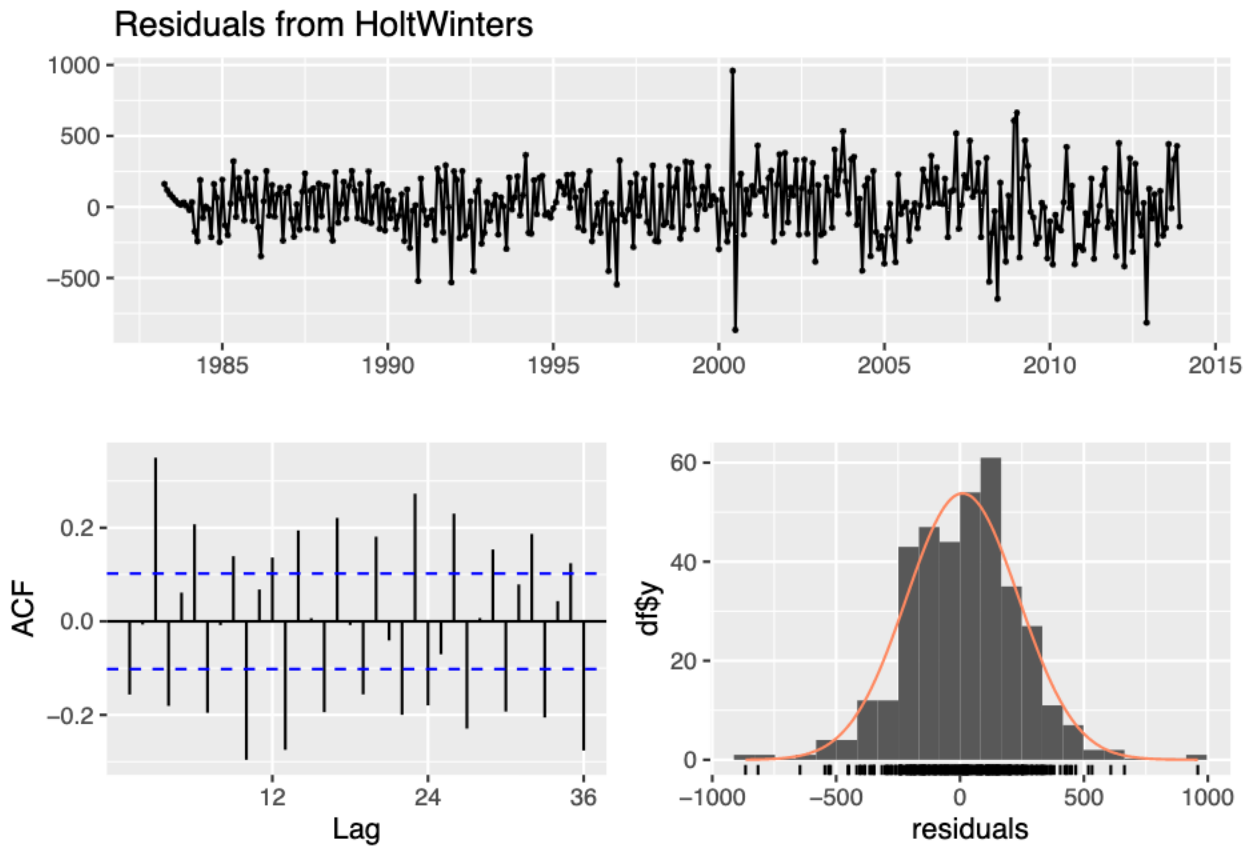
```
## [1] 227.4637
```

```
# Determine the best model
best_model <- if (rmse_hw < rmse_damped) hw_model else hw_damped

# Extract residuals
residuals_best <- residuals(best_model)

# Check if residuals are valid
```

```r
if (is.null(residuals_best)) {
  stop("Residuals could not be extracted from the best model.")
}

# Plot residuals using base R
plot(residuals_best, main = "Residuals from Best Model",
     ylab = "Residuals", xlab = "Time", col = "blue")
abline(h = 0, col = "red", lty = 2)
```



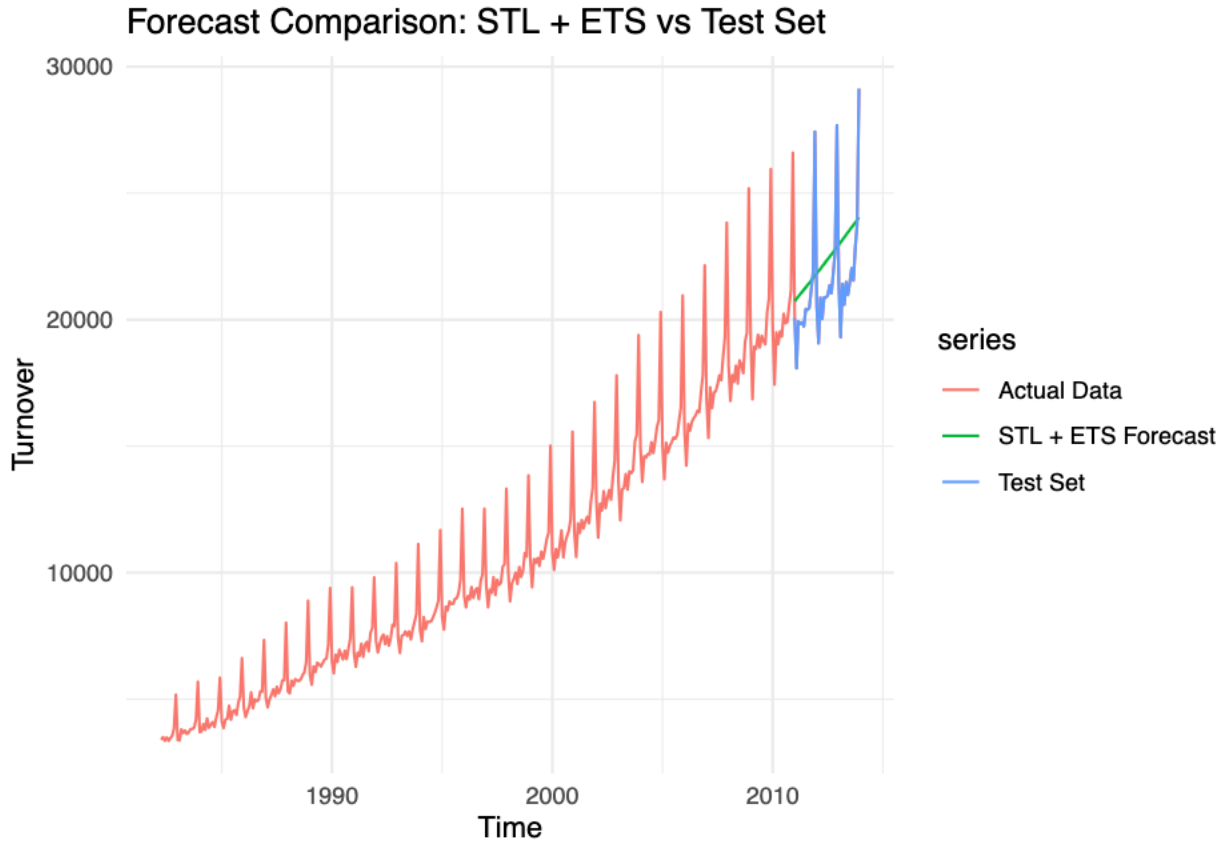

```r
# Use checkresiduals for diagnostics
checkresiduals(best_model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from HoltWinters
## Q* = 307.11, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

```r
## STL Decomposition with ETS on Seasonally Adjusted Data
library(forecast)
library(ggplot2)

# Define the cutoff for training and test sets
cutoff <- c(2010, 12)  # Example: up to December 2010
train_set <- window(turnover_ts, end = cutoff)
test_set <- window(turnover_ts, start = c(2011, 1))

# Box-Cox Transformation
lambda <- BoxCox.lambda(train_set)  # Optimal lambda for training set
train_boxcox <- BoxCox(train_set, lambda)

# STL Decomposition
stl_fit <- stl(train_boxcox, s.window = "periodic")

# Seasonally Adjusted Data
```

```r
seasonally_adjusted <- seasadj(stl_fit)

# Fit ETS Model on Seasonally Adjusted Data
ets_model <- ets(seasonally_adjusted)

# Forecasting on the Test Set Horizon
ets_forecast <- forecast(ets_model, h = length(test_set))

# Reverse Box-Cox Transformation for Forecasts
ets_forecast$mean <- InvBoxCox(ets_forecast$mean, lambda)
ets_forecast$lower <- InvBoxCox(ets_forecast$lower, lambda)
ets_forecast$upper <- InvBoxCox(ets_forecast$upper, lambda)

# Calculate RMSE
rmse_ets <- sqrt(mean((test_set - ets_forecast$mean)^2, na.rm = TRUE))

# Visualize the Results
autoplot(turnover_ts, series = "Actual Data") +
  autolayer(ets_forecast$mean, series = "STL + ETS Forecast") +
  autolayer(test_set, series = "Test Set") +
  ggtitle("Forecast Comparison: STL + ETS vs Test Set") +
  ylab("Turnover") + xlab("Time") +
  theme_minimal()
```



```r
# Output RMSE
cat("RMSE of STL + ETS model: ", rmse_ets, "\n")
```

```
## RMSE of STL + ETS model:  2187.314
```

Problem 7.11

```r
# library(fpp2)
#
# # Load and plot data
# data("visitors")
# autoplot(visitors) +
#   ggtitle("Monthly Australian Short-Term Overseas Visitors") +
#   ylab("Visitors (in thousands)") +
#   xlab("Year") +
#   theme_minimal()
#
# # Split data into training (up to Apr 2003) and test sets (May 2003-Apr 2005)
# train_set <- window(visitors, end = c(2003, 4))
# test_set <- window(visitors, start = c(2003, 5))
#
# # Check data
# autoplot(train_set) +
#   autolayer(test_set, series = "Test Set", PI = FALSE) +
#   ggtitle("Training and Test Sets") +
#   ylab("Visitors (in thousands)") +
#   xlab("Year") +
#   theme_minimal()
#
# # Fit Holt-Winters multiplicative model
# hw_model <- hw(train_set, seasonal = "multiplicative")
# hw_forecast <- forecast(hw_model, h = length(test_set))
#
# # Forecast and plot Holt-Winters
# autoplot(visitors) +
#   autolayer(hw_forecast$mean, series = "HW Forecast", PI = FALSE) +
#   autolayer(test_set, series = "Test Set", PI = FALSE) +
#   ggtitle("Holt-Winters Multiplicative Forecast") +
#   ylab("Visitors (in thousands)") +
#   xlab("Year") +
#   theme_minimal()
#
# # Calculate RMSE
# rmse_hw <- sqrt(mean((test_set - hw_forecast$mean)^2, na.rm = TRUE))
# cat("RMSE (Holt-Winters Multiplicative):", rmse_hw, "\n")
#
# # Automatic ETS model
# ets_model <- ets(train_set)
# ets_forecast <- forecast(ets_model, h = length(test_set))
#
```

```
# # RMSE for ETS
# rmse_ets <- sqrt(mean((test_set - ets_forecast$mean)^2, na.rm = TRUE))
# cat("RMSE (ETS):", rmse_ets, "\n")
#
# # Box-Cox Transformation
# lambda <- BoxCox.lambda(train_set)
# train_boxcox <- BoxCox(train_set, lambda)
#
# # Fit ETS model with Box-Cox transformation
# ets_boxcox_model <- ets(train_boxcox)
# ets_boxcox_forecast <- forecast(ets_boxcox_model, h = length(test_set))
#
# # Reverse Box-Cox Transformation
# ets_boxcox_forecast$mean <- InvBoxCox(ets_boxcox_forecast$mean, lambda)
#
# # RMSE for ETS with Box-Cox
# rmse_ets_boxcox <- sqrt(mean((test_set - ets_boxcox_forecast$mean)^2, na.rm = TRUE))
# cat("RMSE (ETS with Box-Cox):", rmse_ets_boxcox, "\n")
#
# # Seasonal Naïve Forecast
# snaive_forecast <- snaive(train_set, h = length(test_set))
#
# # RMSE for Seasonal Naïve
# rmse_snaive <- sqrt(mean((test_set - snaive_forecast$mean)^2, na.rm = TRUE))
# cat("RMSE (Seasonal Naïve):", rmse_snaive, "\n")
#
# # STL Decomposition
# stl_fit <- stl(BoxCox(train_set, lambda), s.window = "periodic")
#
# # Seasonally Adjusted Data
# seasonally_adjusted <- seasadj(stl_fit)
#
# # ETS Model on Seasonally Adjusted Data
# stl_ets_model <- ets(seasonally_adjusted)
# stl_ets_forecast <- forecast(stl_ets_model, h = length(test_set))
#
# # Reverse Box-Cox Transformation
# stl_ets_forecast$mean <- InvBoxCox(stl_ets_forecast$mean, lambda)
#
# # RMSE for STL + ETS
# rmse_stl_ets <- sqrt(mean((test_set - stl_ets_forecast$mean)^2, na.rm = TRUE))
# cat("RMSE (STL + ETS):", rmse_stl_ets, "\n")
#
# # Print all RMSE values
# rmse_values <- data.frame(
#   Method = c("Holt-Winters", "ETS", "ETS + Box-Cox", "Seasonal Naive", "STL + ETS"),
#   RMSE = c(rmse_hw, rmse_ets, rmse_ets_boxcox, rmse_snaive, rmse_stl_ets)
# )
#
# print(rmse_values)
#
# # Best model
# best_model <- rmse_values[which.min(rmse_values$RMSE), ]
```

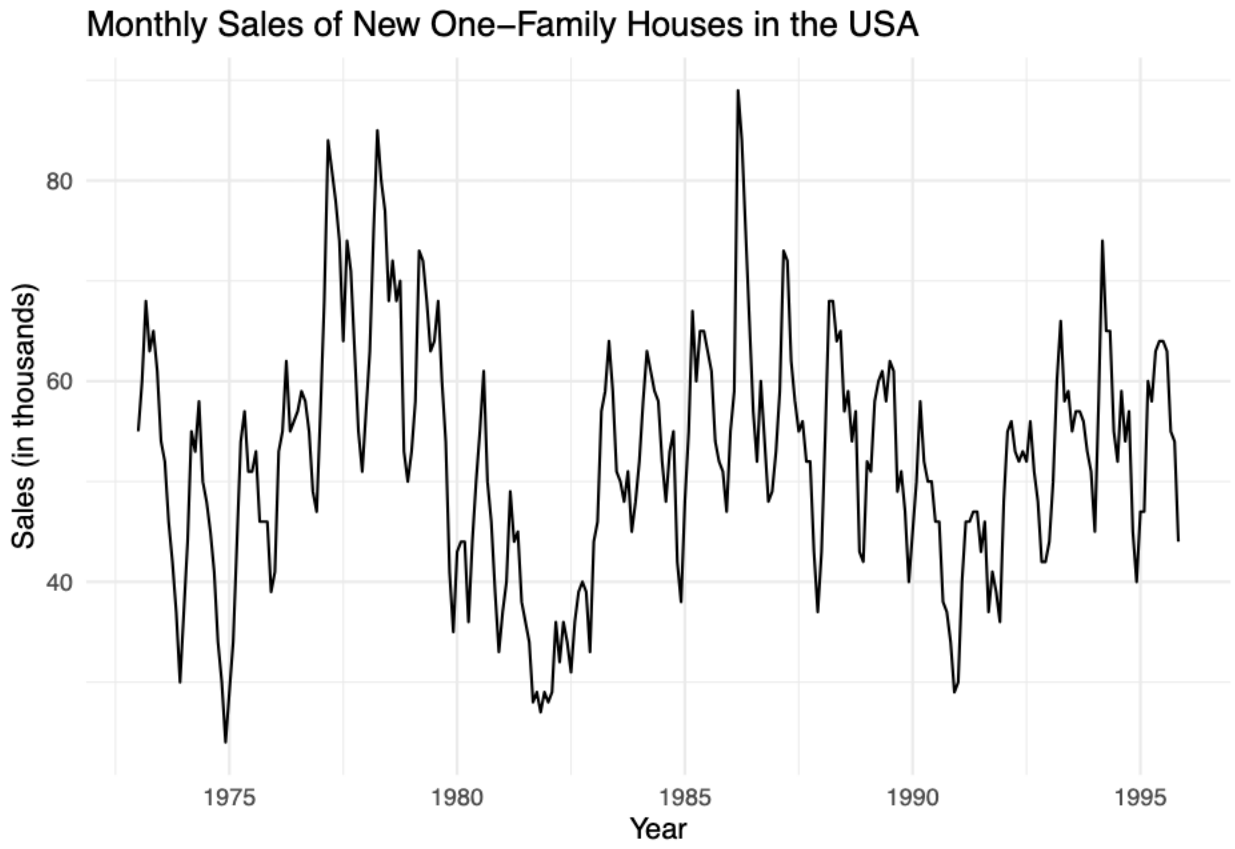
```
# cat("Best Model: ", best_model$Method, " with RMSE: ", best_model$RMSE, "\n")
#
# # Compute residuals for all models
# residuals_hw <- residuals(hw_model)
# residuals_ets <- residuals(ets_model)
# residuals_ets_boxcox <- residuals(ets_boxcox_model)
# residuals_snaive <- residuals(snaive_forecast)
# residuals_stl_ets <- residuals(stl_ets_model)
#
# # Combine residuals into a list
# all_residuals <- list(
#   "Holt-Winters" = residuals_hw,
#   "ETS" = residuals_ets,
#   "ETS + Box-Cox" = residuals_ets_boxcox,
#   "Seasonal Naïve" = residuals_snaive,
#   "STL + ETS" = residuals_stl_ets
# )
#
# # Plot residuals for each model
# par(mfrow = c(3, 2)) # Arrange plots in a grid
# for (method in names(all_residuals)) {
#   if (!is.null(all_residuals[[method]])) {
#     plot(all_residuals[[method]], main = paste("Residuals:", method), ylab = "Residuals")
#     abline(h = 0, col = "red", lty = 2)
#   }
# }
# par(mfrow = c(1, 1)) # Reset plotting layout
#
# # Perform residual diagnostics for each model
# cat("Residual Diagnostics:\n")
# for (method in names(all_residuals)) {
#   if (!is.null(all_residuals[[method]])) {
#     cat("\n", method, ":\n")
#     checkresiduals(all_residuals[[method]], plot = FALSE) # Print diagnostics without plotting
#   }
# }
#
# # Use tsCV for time series cross-validation
# cv_rmse <- function(forecast_function, h) {
#   sqrt(mean(tsCV(visitors, forecast_function, h = h)^2, na.rm = TRUE))
# }
#
# # Compute RMSE for each method using cross-validation
# cv_hw <- cv_rmse(function(x, h) hw(x, h = h, seasonal = "multiplicative"), h = 24)
# cv_ets <- cv_rmse(function(x, h) forecast(ets(x), h = h), h = 24)
# cv_snaive <- cv_rmse(function(x, h) snaive(x, h = h), h = 24)
# cv_stl_ets <- cv_rmse(function(x, h) {
#   lambda <- BoxCox.lambda(x)
#   seasadj_fit <- seasadj(stl(BoxCox(x, lambda), s.window = "periodic"))
#   forecast(ets(seasadj_fit), h = h)$mean
# }, h = 24)
#
# # Print cross-validation results
```

```r
# cv_results <- data.frame(
#   Method = c("Holt-Winters", "ETS", "Seasonal Naïve", "STL + ETS"),
#   CV_RMSE = c(cv_hw, cv_ets, cv_snaive, cv_stl_ets)
# )
#
# print(cv_results)
```

```r
library(fpp2)        # For time series data and utilities
```

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

## v fma       2.5     v expsmooth 2.3

##
```

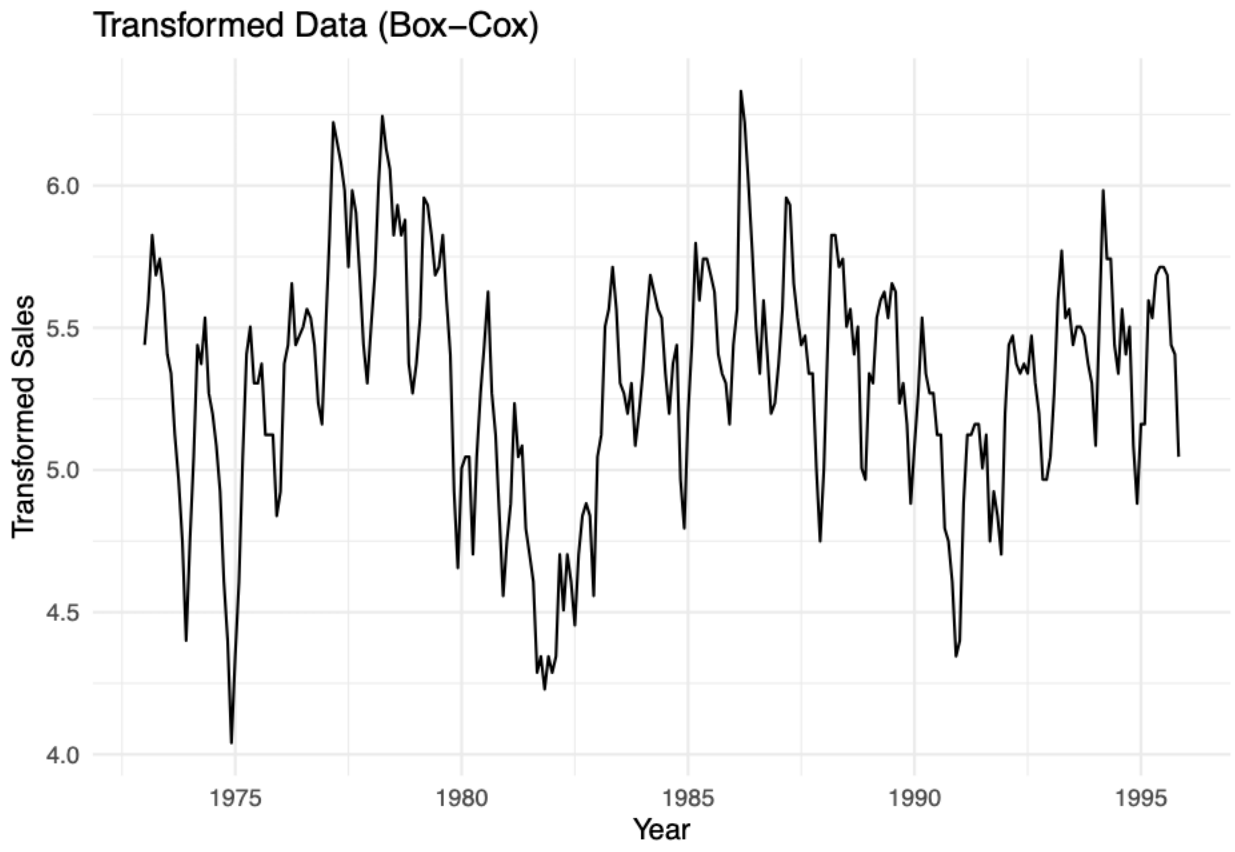
```r
library(tseries)     # For stationarity tests
library(forecast)    # For ARIMA and ETS modeling

# Load and plot the data
data("hsales")
autoplot(hsales) +
  ggtitle("Monthly Sales of New One-Family Houses in the USA") +
  ylab("Sales (in thousands)") +
  xlab("Year") +
  theme_minimal()
```



```r
# 1. Transformation (Box-Cox Test for Normality)
lambda <- BoxCox.lambda(hsales)
cat("Optimal Lambda (Box-Cox):", lambda, "\n")
```

```
## Optimal Lambda (Box-Cox): 0.1454608
```

```r
# Apply Box-Cox transformation if needed
hsales_transformed <- BoxCox(hsales, lambda)

autoplot(hsales_transformed) +
  ggtitle("Transformed Data (Box-Cox)") +
  ylab("Transformed Sales") +
  xlab("Year") +
  theme_minimal()
```

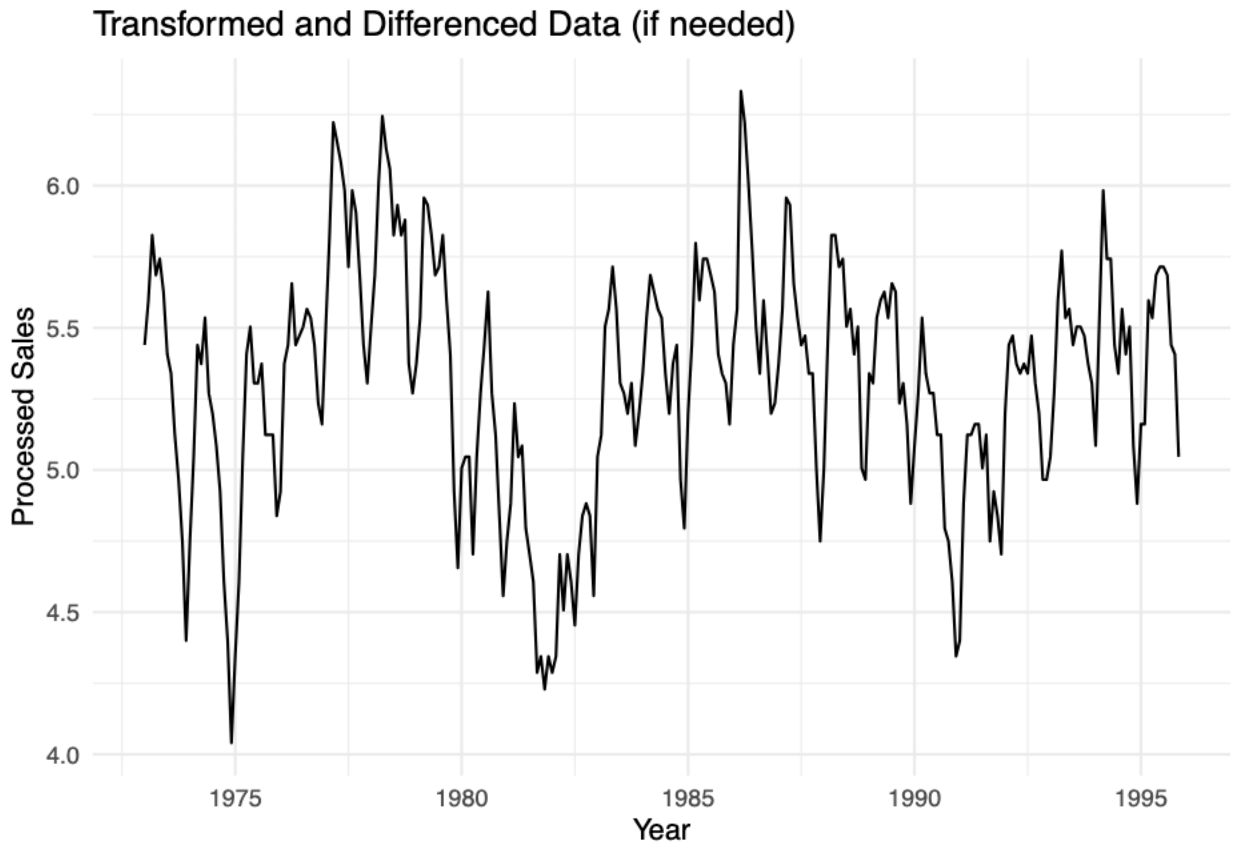



```r
# 2. Stationarity Check (ADF Test)
adf_test <- adf.test(hsales_transformed)
cat("ADF Test p-value (Original Data):", adf_test$p.value, "\n")
```

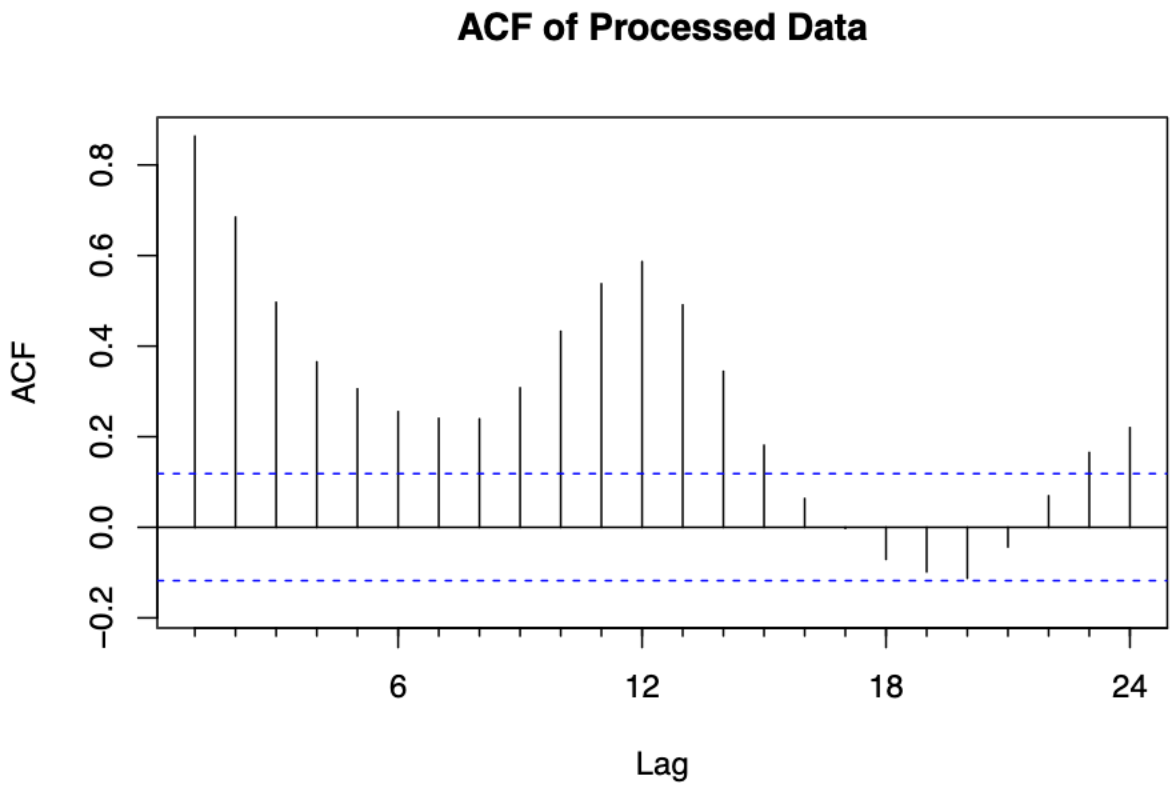
```
## ADF Test p-value (Original Data): 0.01933394
```

```r
# Differencing if not stationary
if (adf_test$p.value > 0.05) {
```

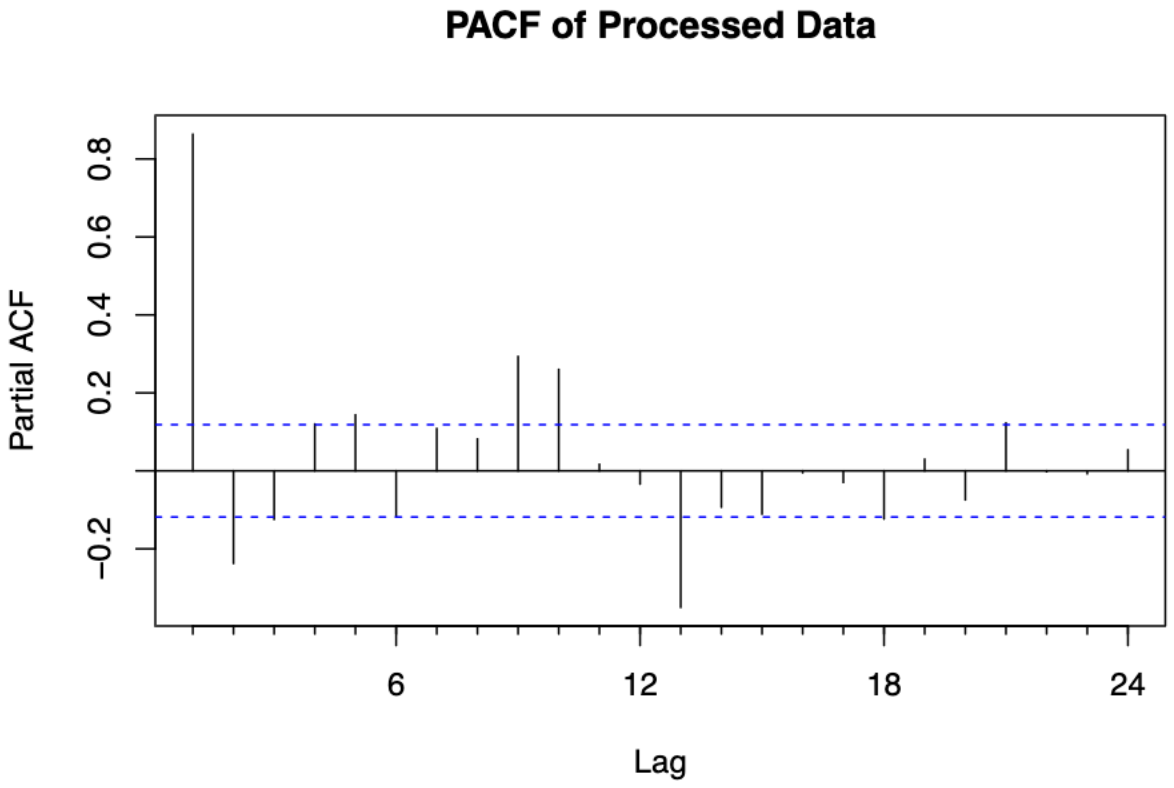
```r
  hsales_diff <- diff(hsales_transformed, differences = 1)
  adf_test_diff <- adf.test(hsales_diff)
  cat("ADF Test p-value (First Differenced Data):", adf_test_diff$p.value, "\n")
  hsales_final <- hsales_diff
} else {
  hsales_final <- hsales_transformed
}

autoplot(hsales_final) +
  ggtitle("Transformed and Differenced Data (if needed)") +
  ylab("Processed Sales") +
  xlab("Year") +
  theme_minimal()
```




```r
# 3. Identify ARIMA Models
Acf(hsales_final, main = "ACF of Processed Data")
```




```
Pacf(hsales_final, main = "PACF of Processed Data")
```



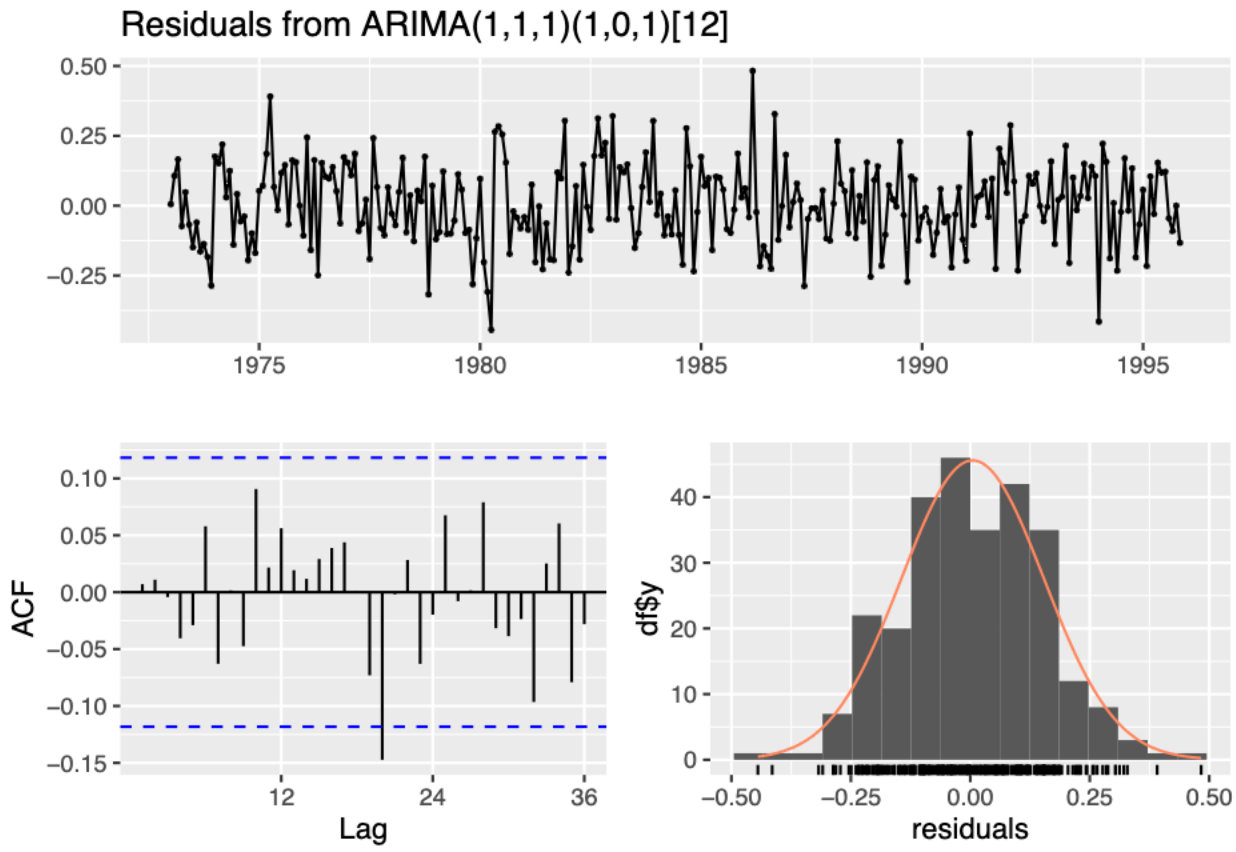

```
# Fit a couple of ARIMA models
arima1 <- Arima(hsales_transformed, order = c(1, 1, 1), seasonal = c(1, 0, 1))
arima2 <- Arima(hsales_transformed, order = c(2, 1, 2), seasonal = c(1, 0, 1))

# 4. Compare AIC values
aic_values <- data.frame(
  Model = c("ARIMA(1,1,1)(1,0,1)", "ARIMA(2,1,2)(1,0,1)"),
  AIC = c(AIC(arima1), AIC(arima2))
)
print(aic_values)
```

```
##                 Model       AIC
## 1 ARIMA(1,1,1)(1,0,1) -222.2691
## 2 ARIMA(2,1,2)(1,0,1) -218.3937
```

```
# Choose the best model
best_arima <- if (AIC(arima1) < AIC(arima2)) arima1 else arima2
cat("Best ARIMA Model Order:", best_arima$arma, "\n")
```

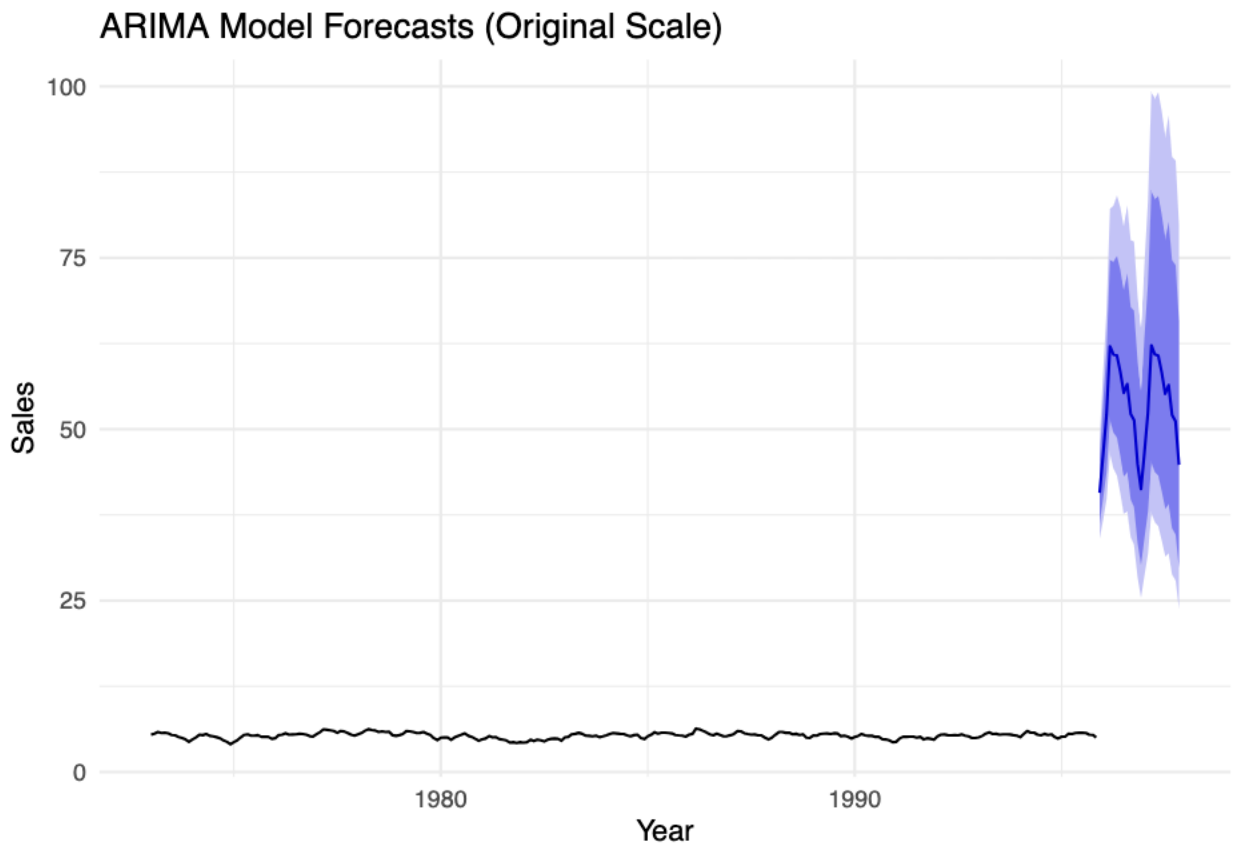
```
## Best ARIMA Model Order: 1 1 1 1 12 1 0
```

```
# 5. Diagnostics
checkresiduals(best_arima)
```

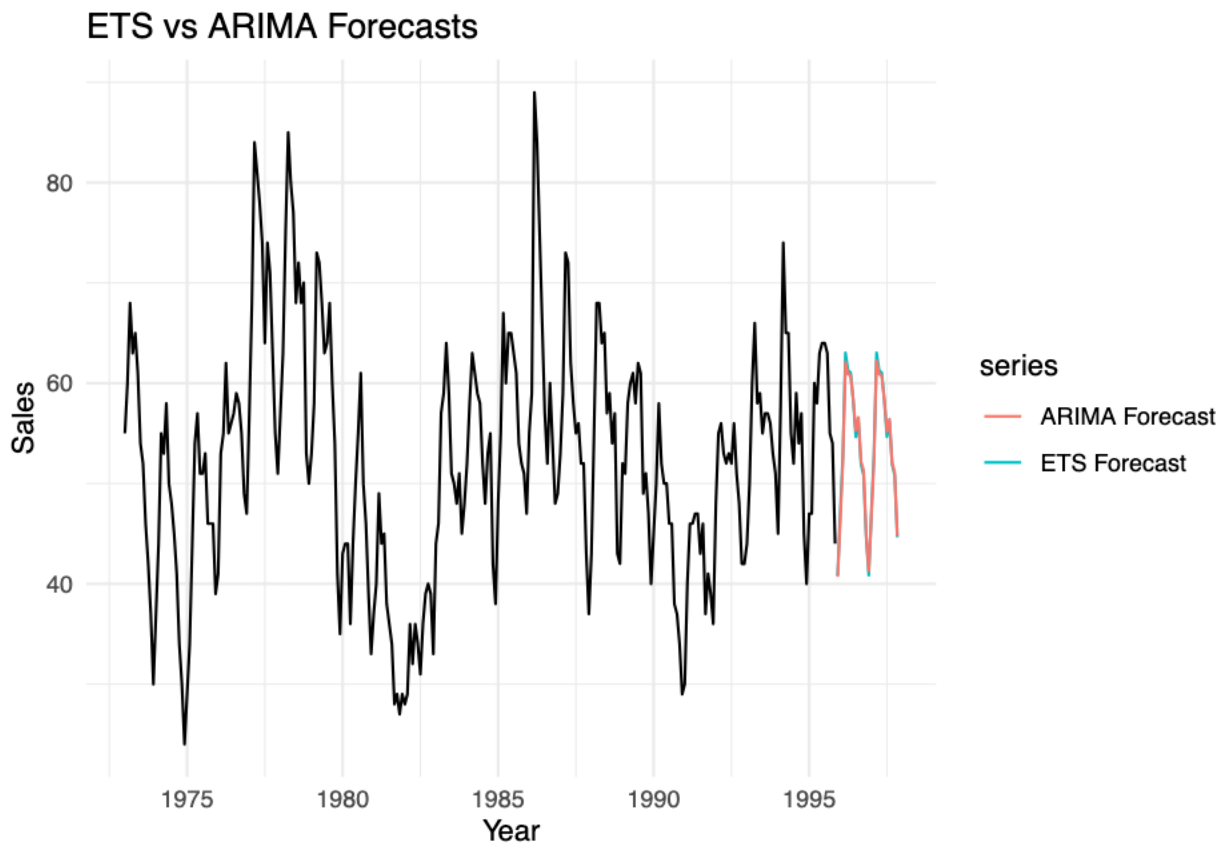



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(1,0,1)[12]
## Q* = 17.923, df = 20, p-value = 0.5925
##
## Model df: 4.   Total lags used: 24
```

```r
# 6. Forecast 24 Months
arima_forecast <- forecast(best_arima, h = 24)

# Inverse Box-Cox Transformation for forecasts
arima_forecast$mean <- InvBoxCox(arima_forecast$mean, lambda)
arima_forecast$lower <- InvBoxCox(arima_forecast$lower, lambda)
arima_forecast$upper <- InvBoxCox(arima_forecast$upper, lambda)

autoplot(arima_forecast) +
  ggtitle("ARIMA Model Forecasts (Original Scale)") +
  ylab("Sales") +
  xlab("Year") +
  theme_minimal()
```




```r
# 7. ETS Model for Comparison
ets_model <- ets(hsales)
ets_forecast <- forecast(ets_model, h = 24)

# Plot comparison of ETS and ARIMA forecasts
autoplot(hsales) +
  autolayer(ets_forecast$mean, series = "ETS Forecast") +
  autolayer(arima_forecast$mean, series = "ARIMA Forecast") +
  ggtitle("ETS vs ARIMA Forecasts") +
  ylab("Sales") +
  xlab("Year") +
  theme_minimal()
```




```
# Compare RMSE
actual_data <- tail(hsales, 24)  # Adjust to real forecast comparison range
rmse_arima <- sqrt(mean((actual_data - arima_forecast$mean)^2, na.rm = TRUE))
```

```
## Warning in .cbind.ts(list(e1, e2), c(deparse(substitute(e1))[1L],
## deparse(substitute(e2))[1L]), : non-intersecting series
```

```
rmse_ets <- sqrt(mean((actual_data - ets_forecast$mean)^2, na.rm = TRUE))
```

```
## Warning in .cbind.ts(list(e1, e2), c(deparse(substitute(e1))[1L],
## deparse(substitute(e2))[1L]), : non-intersecting series
```

```
cat("RMSE (ARIMA):", rmse_arima, "\n")
```

```
## RMSE (ARIMA): NaN
```

```
cat("RMSE (ETS):", rmse_ets, "\n")
```

```
## RMSE (ETS): NaN
```