ETS Model

Yifan Han

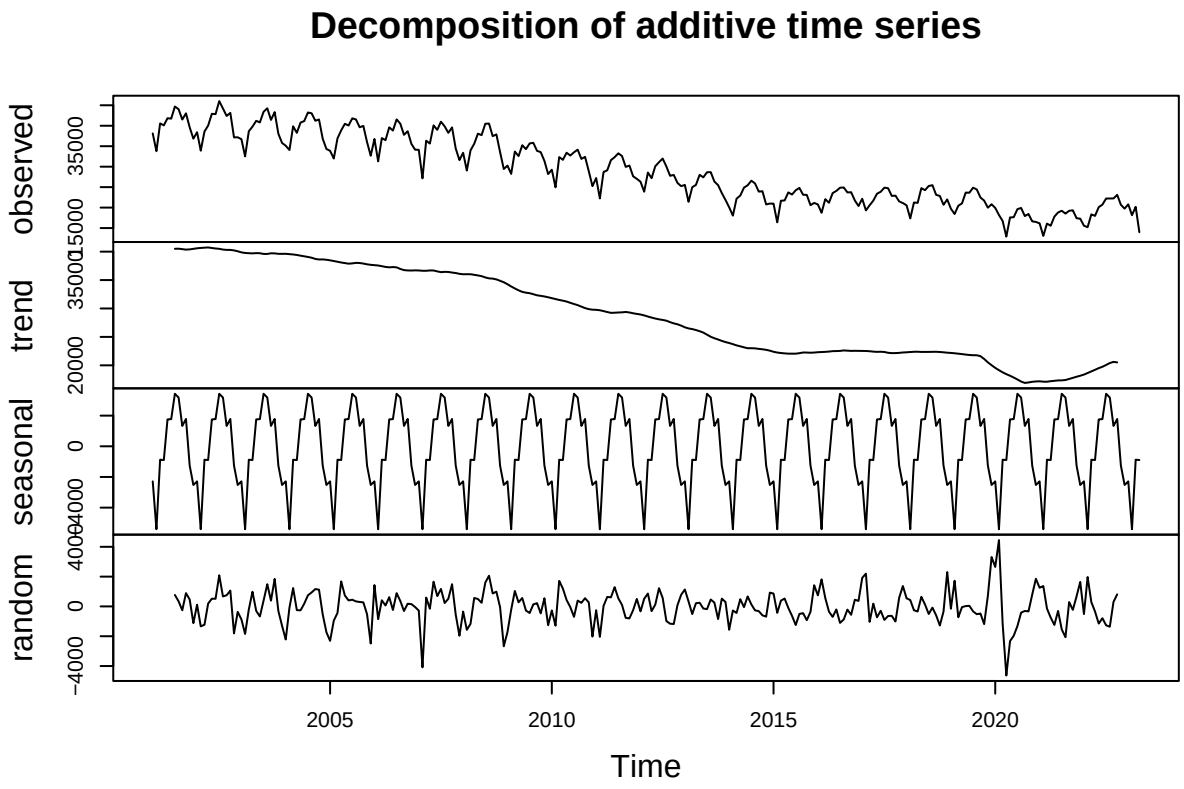
2024-05-23

Data Preparation

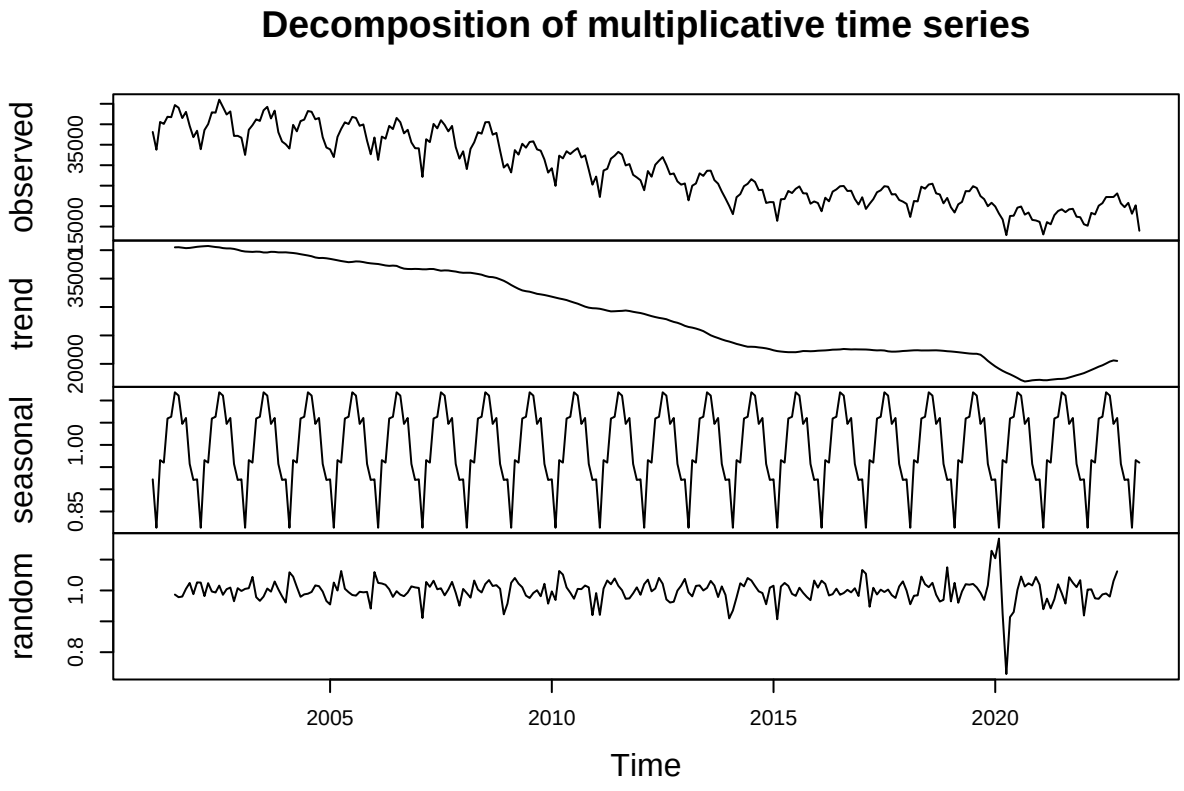
```
# Load data
data <- read.csv('/Users/yifan_/Desktop/Classes/Time series/final/crime_data.csv')

# Convert to time series data
ts_data <- ts(data[, c("total_cases")], start=c(2001, 1), frequency=12)

# Define training and test sets
train_end <- c(2021, 12)
test_start <- c(2022, 1)
test_end <- c(2022, 12)
ts_train <- window(ts_data, end=train_end)
ts_test <- window(ts_data, start=test_start, end=test_end)
```

Seasonal decomposition

```
# Perform seasonal decomposition with an additive model
decomp_additive <- decompose(ts_data, type="additive")

# Perform seasonal decomposition with a multiplicative model
decomp_multiplicative <- decompose(ts_data, type="multiplicative")

# Plot the decompositions
par(mfrow=c(2, 1))
plot(decomp_additive)
```




```
plot(decomp_multiplicative)
```




```
# Determine the type of seasonality
additive_std <- sd(decomp_additive$seasonal, na.rm=TRUE)
multiplicative_std <- sd(decomp_multiplicative$seasonal, na.rm=TRUE)
```

```r
seasonality_type <- ifelse(additive_std < multiplicative_std, "additive", "multiplicative")
print(paste("The seasonality type is:", seasonality_type))
```

```
## [1] "The seasonality type is: multiplicative"
```

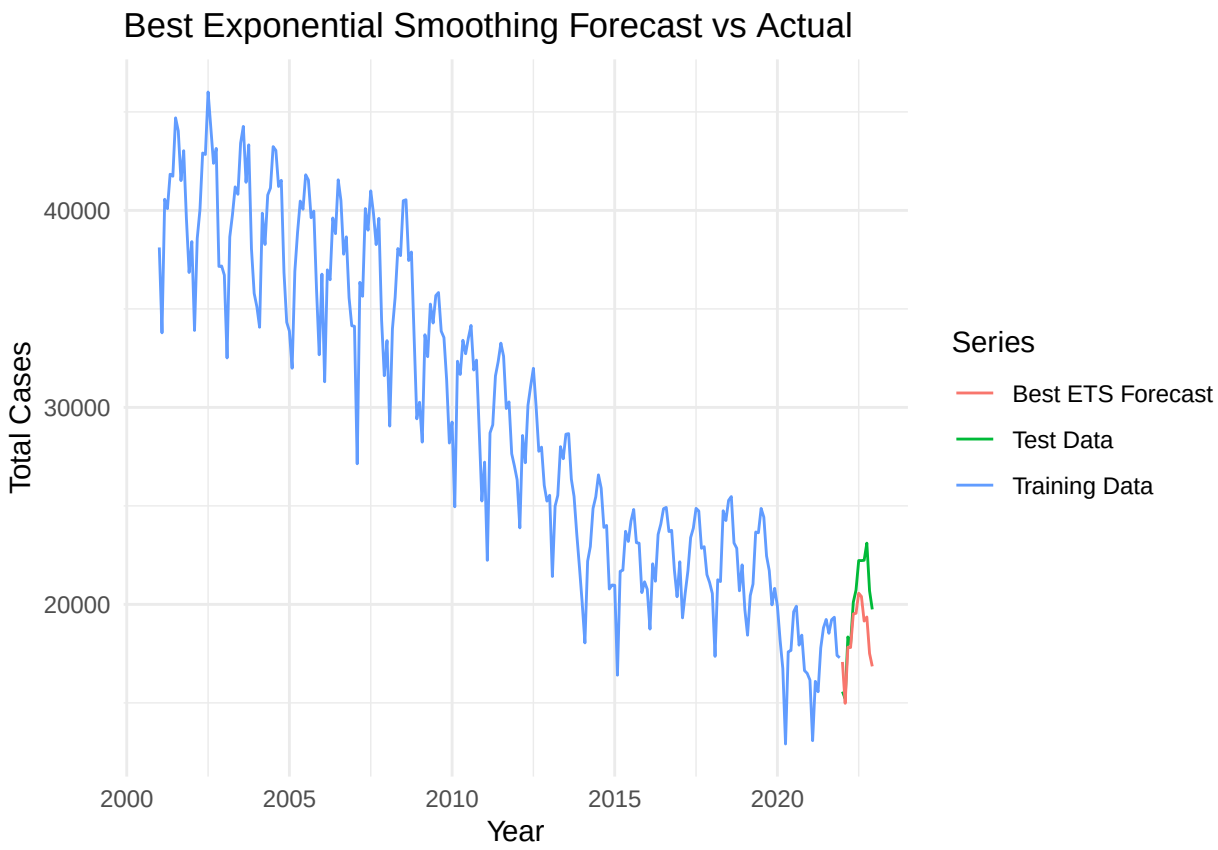
Model

```r
ets_models <- list( "MAM", "MMM")

# Initialize lists to store errors and metrics
error_sliding_list <- list()
error_expanding_list <- list()
rmse_sliding <- list()
smape_sliding <- list()
rmse_expanding <- list()
smape_expanding <- list()

# Sliding window and expanding window cross-validation
k <- 72
p <- 12
st <- 2014 + 11/12

for (model_type in ets_models) {
  error_sliding <- matrix(NA, 12, 12)
  error_expanding <- matrix(NA, 12, 12)
  smape_sliding_temp <- numeric(12)
  smape_expanding_temp <- numeric(12)

  for(i in 1:12) {
    train <- window(ts_data, start=st+(i-k+1)/p, end=st+i/p)  # Sliding window
    train2 <- window(ts_data, start=c(2008, 2), end=st+i/p)   # Expanding window
    test <- window(ts_data, start=st + (i+1)/p, end=st+(i+12)/p)

    model1 <- tryCatch(ets(train, model=model_type), error=function(e) NULL)
    model2 <- tryCatch(ets(train2, model=model_type), error=function(e) NULL)

    if (!is.null(model1)) {
      f1 <- forecast(model1, h=12)
      error_sliding[i, 1:length(test)] <- f1[['mean']] - test
      smape_sliding_temp[i] <- smape(f1[['mean']], test)
    }

    if (!is.null(model2)) {
      f2 <- forecast(model2, h=12)
      error_expanding[i, 1:length(test)] <- f2[['mean']] - test
      smape_expanding_temp[i] <- smape(f2[['mean']], test)
    }
  }

  error_sliding_list[[model_type]] <- error_sliding
  error_expanding_list[[model_type]] <- error_expanding

  rmse_sliding[[model_type]] <- sqrt(colMeans(error_sliding^2, na.rm=TRUE))
```

```
  rmse_expanding[[model_type]] <- sqrt(colMeans(error_expanding^2, na.rm=TRUE))

  smape_sliding[[model_type]] <- smape_sliding_temp
  smape_expanding[[model_type]] <- smape_expanding_temp
}

# Compare and select the best model
best_model <- NULL
min_rmse <- Inf

for (model_type in ets_models) {
  avg_rmse_sliding <- mean(rmse_sliding[[model_type]], na.rm=TRUE)
  avg_rmse_expanding <- mean(rmse_expanding[[model_type]], na.rm=TRUE)

  if (avg_rmse_sliding < min_rmse) {
    min_rmse <- avg_rmse_sliding
    best_model <- model_type
  }

  if (avg_rmse_expanding < min_rmse) {
    min_rmse <- avg_rmse_expanding
    best_model <- model_type
  }
}

print(paste("Best model:", best_model))
```

```
## [1] "Best model: MMM"
```

```
# Fit the best model on the entire training data
best_ets_model <- ets(ts_train, model=best_model)
best_ets_forecast <- forecast(best_ets_model, h=length(ts_test))

# Plot the forecasts and include training data
autoplot(ts_train, series="Training Data") +
  autolayer(ts_test, series="Test Data") +
  autolayer(best_ets_forecast$mean, series="Best ETS Forecast") +
  ggtitle("Best Exponential Smoothing Forecast vs Actual") +
  xlab("Year") + ylab("Total Cases") +
  theme_minimal() +
  guides(colour=guide_legend(title="Series"))
```




```r
print(paste("Best RMSE:", min_rmse))
```

```
## [1] "Best RMSE: 1074.47697974318"
```

Comparision

```r
# Create a data frame for RMSE and sMAPE values
performance_comparison <- data.frame(
  Model = c('MAM','MMM'),
  RMSE = c(
    mean(rmse_sliding[["MAM"]], na.rm=TRUE),
    mean(rmse_sliding[["MMM"]], na.rm=TRUE)
  ),
  sMAPE = c(
    mean(smape_sliding[["MAM"]], na.rm=TRUE),
    mean(smape_sliding[["MMM"]], na.rm=TRUE)
  )
)

# Print the performance comparison table
print(performance_comparison)
```

```
##   Model     RMSE      sMAPE
## 1   MAM 1141.045 0.04638236
## 2   MMM 1074.477 0.04341728
```