Assignment 3

Jaewoo Cho

FPP3 8.8 Exercises: 5(a-e)

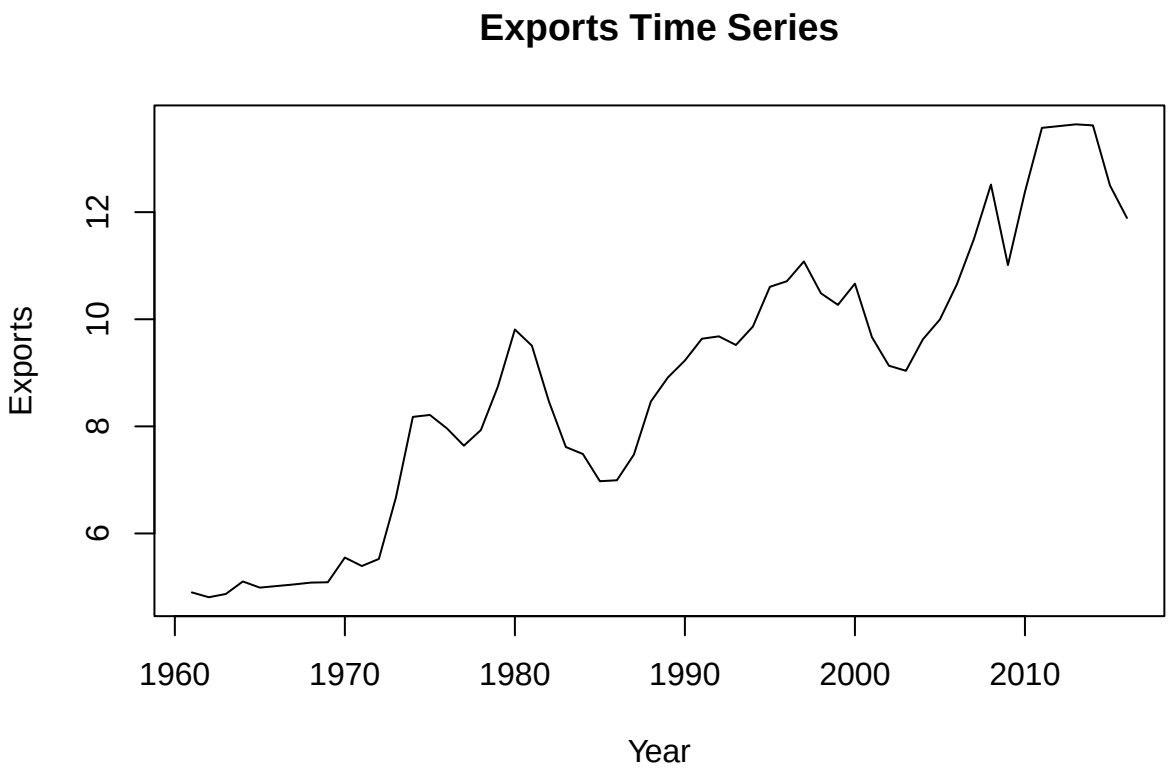
Fit and Forecast Methane Data using ETS

5.

Data set `global_economy` contains the annual Exports from many countries. Select one country to analyze.

a. Plot the Exports series and discuss the main features of the data.

```
global <- global_economy
# Subset data for the United States
us_data <- global_economy[global_economy$Country == "United States", ]
us_data <- na.omit(us_data)
us_data <- us_data %>% select(Year, Exports)
plot(us_data$Year, us_data$Exports, type = "l", xlab = "Year", ylab = "Exports", main = "Exports Time S
```

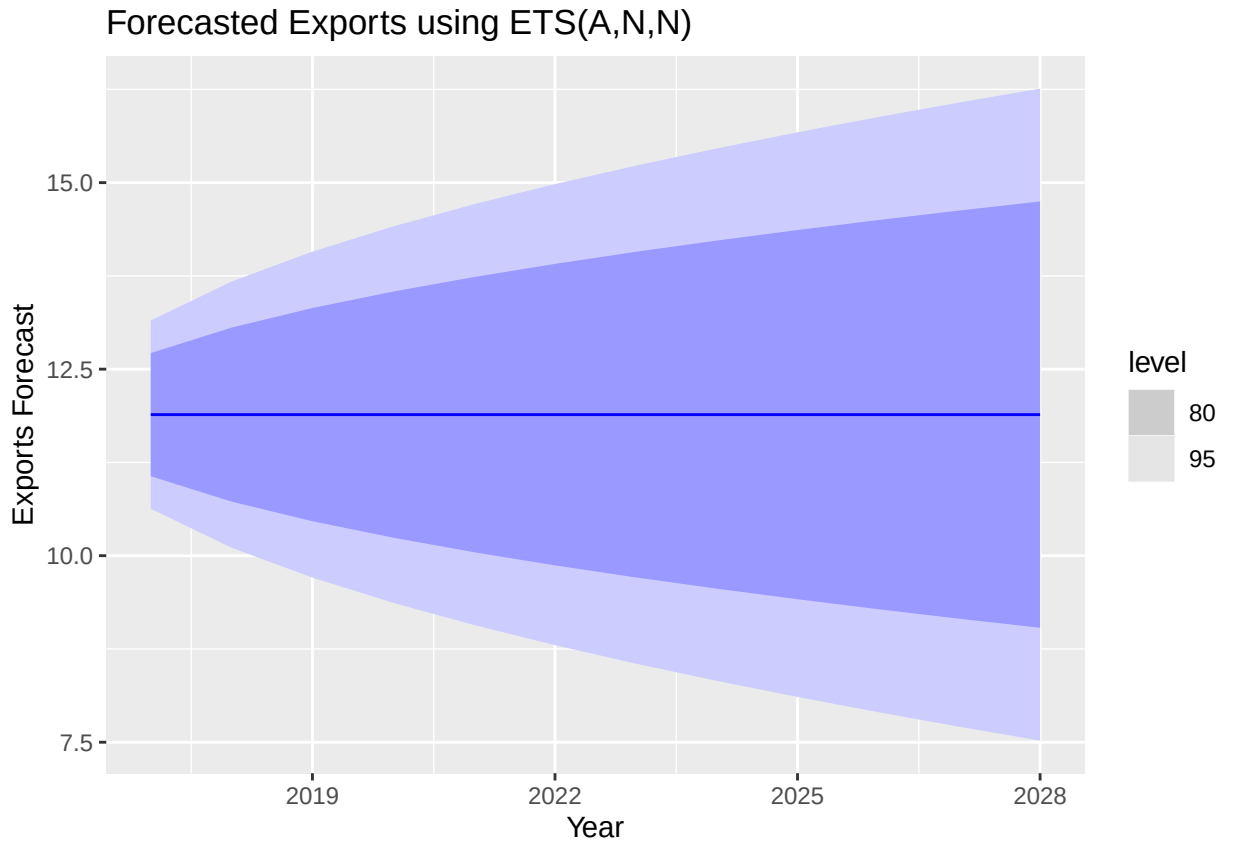


> Main features of the data - The data consists of annual data of Years, Exports, population, and Country that explains the exports of the global economy with trends. > Discuss features of the time series - Seasonality: Many time series data exhibit seasonality, which refers to recurring patterns or cycles at fixed intervals. For example, retail sales often exhibit yearly seasonality with peaks during the holiday season. - Trends: Time series data may show long-term trends, which represent gradual and sustained changes in the data over time. Trends can be upward (increasing) or downward (decreasing). - Cyclic Patterns: Cyclic patterns are different from seasonality in that they represent periodic fluctuations that are not tied to specific calendar intervals. Cycles can be longer or shorter than a year and may not have a fixed duration.

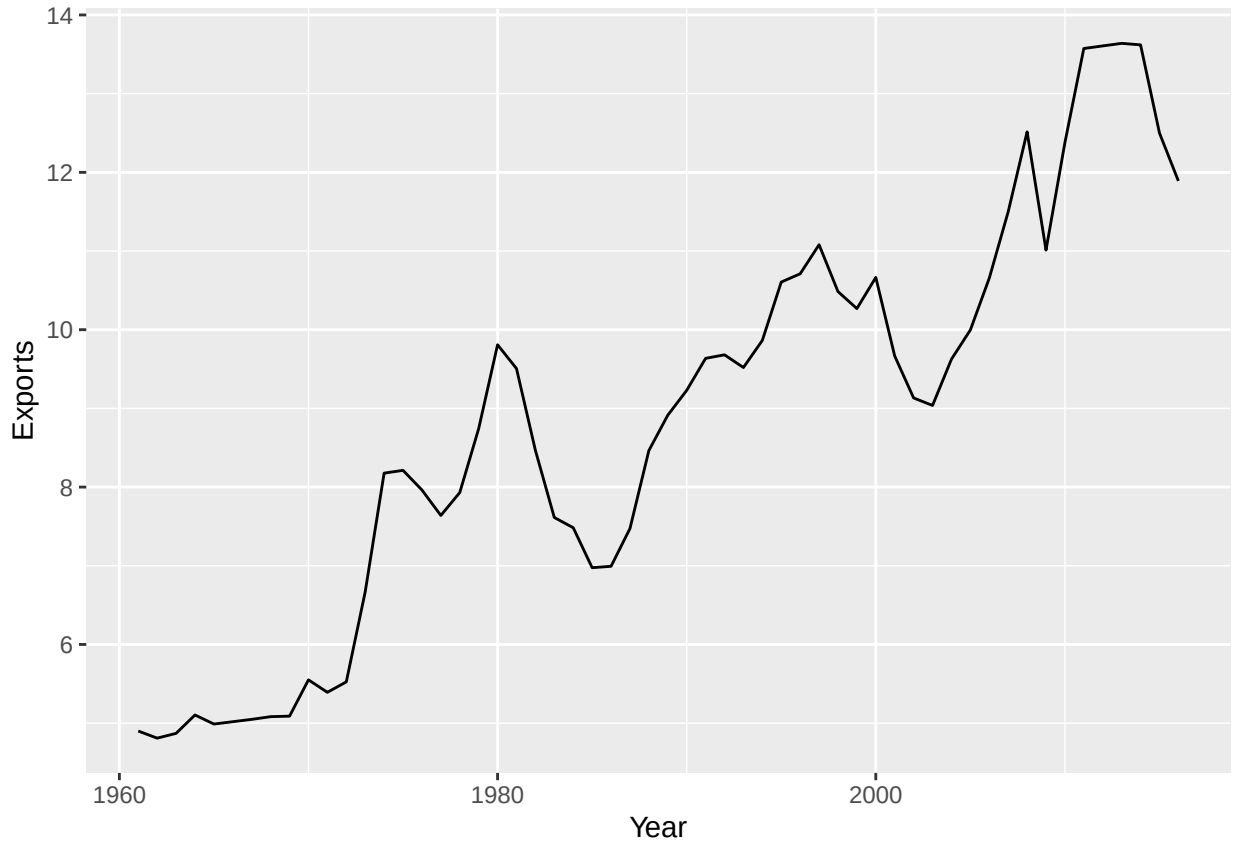
b. Use an ETS(A,N,N) model to forecast the series, and plot the forecasts.

```
# Convert us_data to tsibble
us_data <- us_data %>% as_tsibble(index = Year)

# Fit an ETS(A,N,N) model to the data
us_fit <- us_data %>% model(ETS(Exports ~ error("A") + trend("N") + season("N")))

# Forecast future values
us_forecast <- us_fit %>%
  forecast(h = 12)  # Change the forecast horizon as needed

# Plot the forecasts
autoplot(us_forecast) +
  labs(
    y = "Exports Forecast",
```

```
    x = "Year",
    title = "Forecasted Exports using ETS(A,N,N)"
  )
```



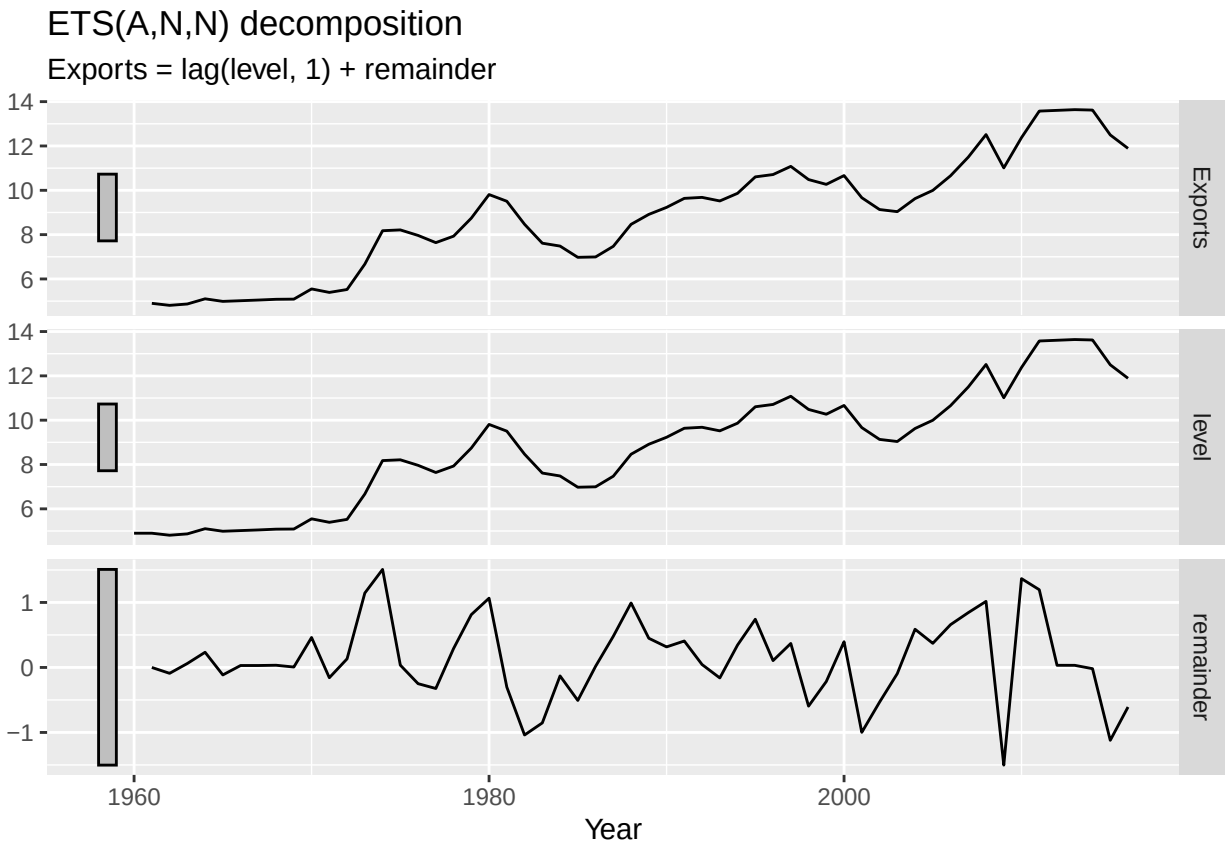

```
us_data_ts <- as_tsibble(us_data, index = Year)
# Plot time series
us_data_ts %>% autoplot(Exports) +labs(y = "Exports",x = "Year")
```



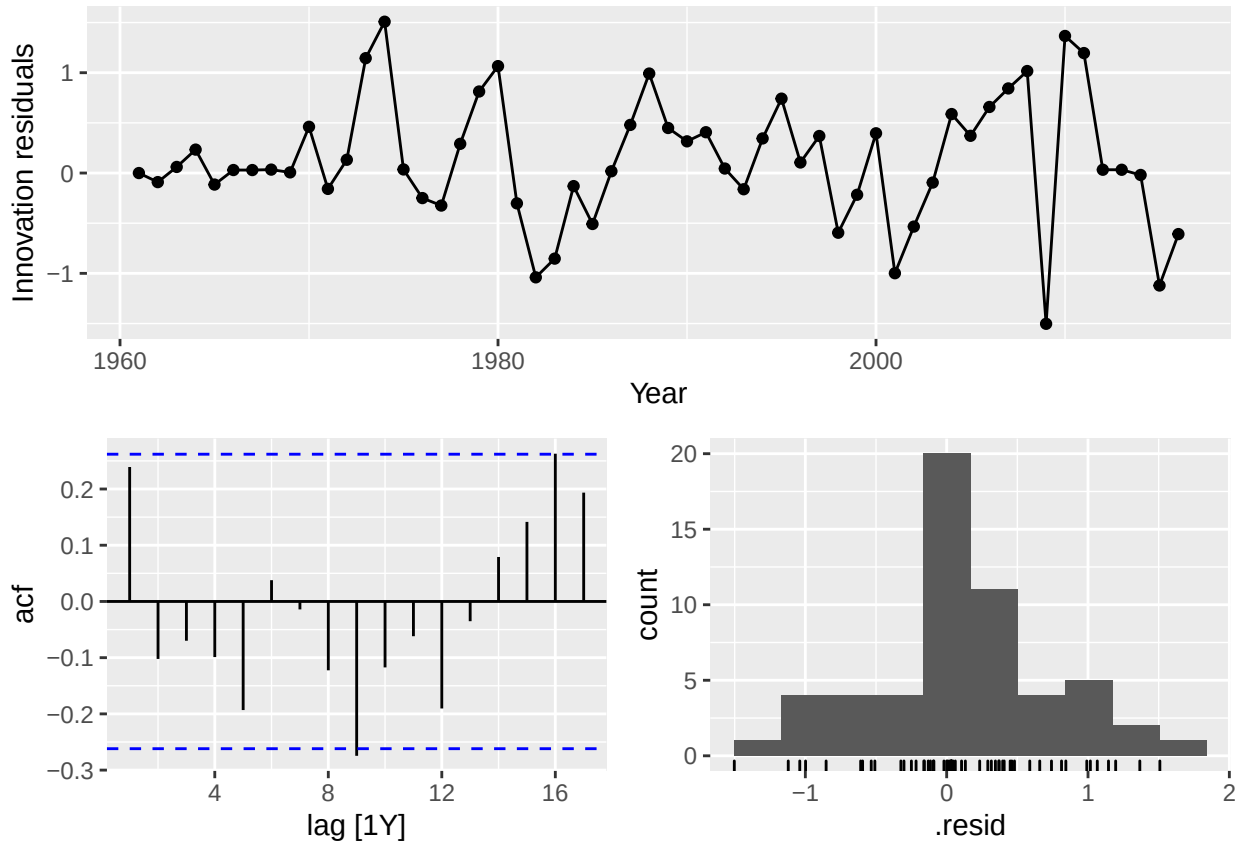

```
# Estimate parameters
fit <- us_data_ts %>% model(ANN = ETS(Exports ~ error("A") + trend("N") + season("N")))

# Report fit
report(fit)
```

```
## Series: Exports
## Model: ETS(A,N,N)
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l[0]
##  4.89991
##
##   sigma^2:  0.4142
##
##      AIC     AICc      BIC
## 180.0245 180.4861 186.1006
```

```
# Plot components
components(fit) %>% autoplot()
```




```
# Plot residuals
fit %>% gg_tsresiduals()
```




```
# Estimate parameters
fit <- us_data_ts %>%
  model(
    ANN = ETS(Exports ~ error("A") + trend("N") + season("N")),
    AAN = ETS(Exports ~ error("A") + trend("A") + season("N")),
    AAdN = ETS(Exports ~ error("A") + trend("Ad") + season("N")),
    AAA = ETS(Exports ~ error("A") + trend("A") + season("A")),
    AAdA = ETS(Exports ~ error("A") + trend("Ad") + season("A"))
  )
# Report fit
report(fit)
```

```
## # A tibble: 3 x 9
##   .model sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE   MAE
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ANN     0.414   -87.0  180.  180.  186. 0.399 1.01  0.468
## 2 AAN     0.415   -86.0  182.  183.  192. 0.386 0.915 0.476
## 3 AAdN    0.436   -86.8  186.  187.  198. 0.397 1.02  0.470
```

```
# The data vector
data <- c(5.633e+11, 6.051e+11, 6.386e+11, 6.858e+11, 7.437e+11, 8.150e+11, 8.617e+11, 9.425e+11,
          1.0199e+12, 1.075884e+12, 1.16777e+12, 1.282449e+12, 1.428549e+12, 1.548825e+12, 1.688923e+
          2.085951e+12, 2.356571e+12, 2.632143e+12, 2.862505e+12, 3.210956e+12, 3.344991e+12, 3.63813
          4.346734e+12, 4.590155e+12, 4.870217e+12, 5.252629e+12, 5.657693e+12, 5.979589e+12, 6.17404
          6.878718e+12, 7.308755e+12, 7.664060e+12, 8.100201e+12, 8.608515e+12, 9.089168e+12, 9.66062
```

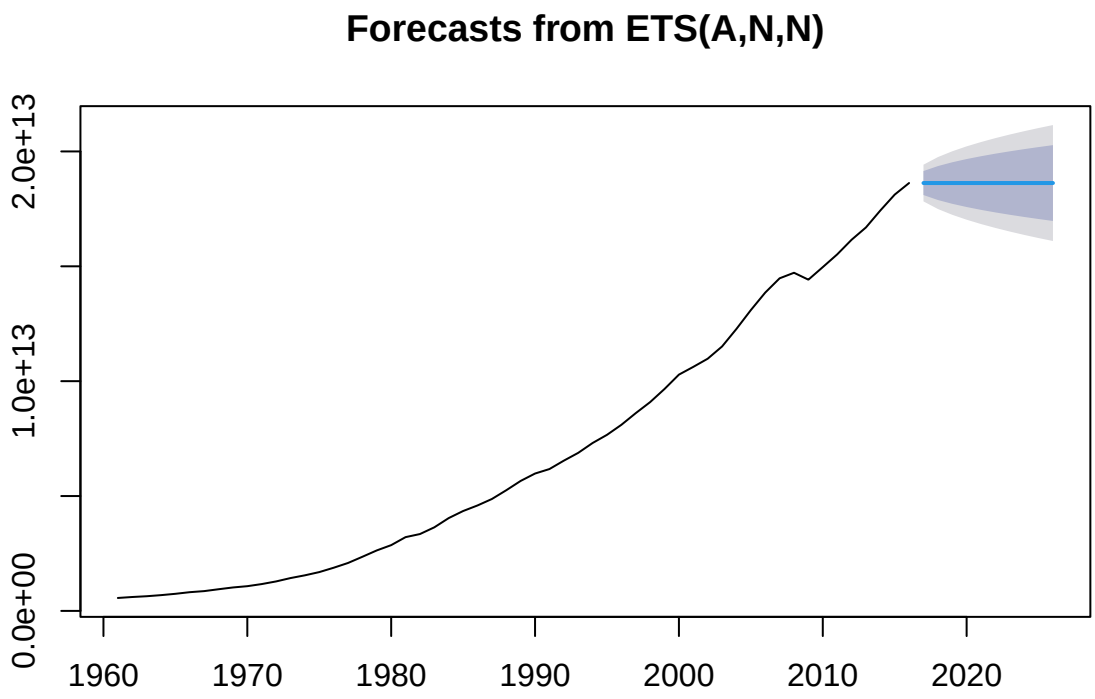
```
            1.062182e+13, 1.097751e+13, 1.151067e+13, 1.227493e+13, 1.309373e+13, 1.385589e+13, 1.44776
            1.441874e+13, 1.496437e+13, 1.551793e+13, 1.615526e+13, 1.669152e+13, 1.742761e+13, 1.81207
```

```
# Creating the ts object
us_data_ts_tsts <- ts(data, start=1961, frequency=1)
```

```
library(forecast)
```

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

```
ets_model <- ets(us_data_ts_tsts, model = "ANN")
forecasted_values <- forecast(ets_model, h = 10)
plot(forecasted_values)
```



c. Compute the RMSE values for the training data.

```
training_data <- window(us_data_ts_tsts, end = c(2010, 1))
fitted_values <- fitted(ets_model)
rmse <- sqrt(mean((training_data - fitted_values)^2))
rmse
```

```
[1] 365452844803
```

d. Compare the results to those from an ETS(A,A,N) model. (Remember that the trended model is using one more parameter than the simpler model.) Discuss the merits of the two forecasting methods for this data set.

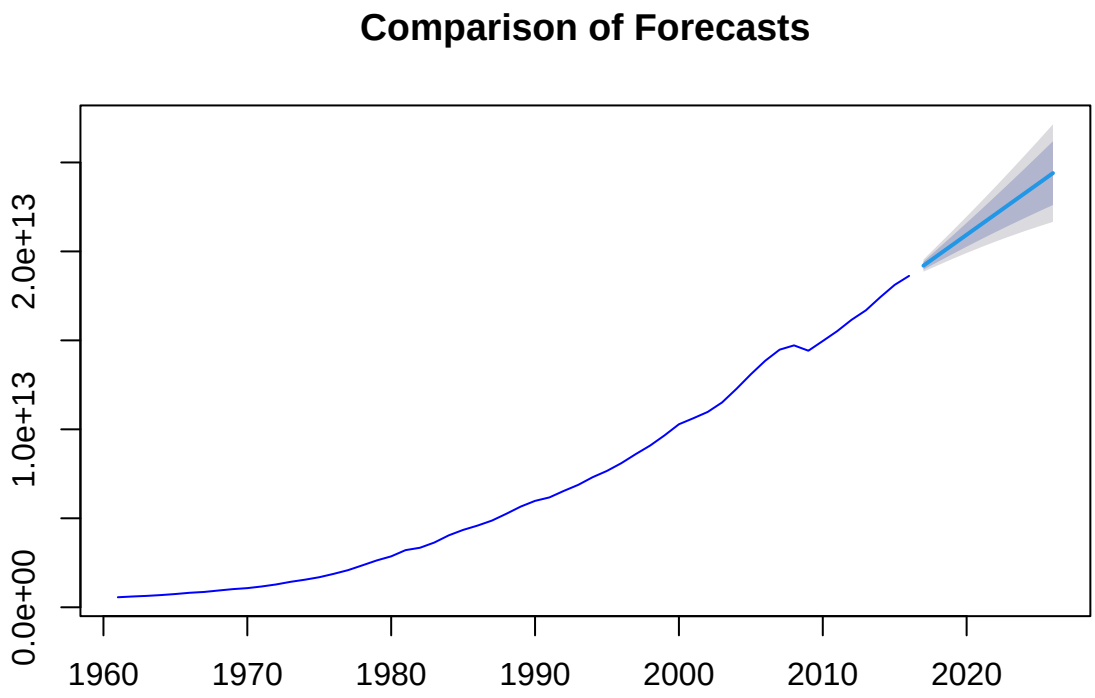
```
ets_aa_model <- ets(us_data_ts_tsts, model = "AAN")
fitted_values_aa <- fitted(ets_aa_model)
rmse_aa <- sqrt(mean((training_data - fitted_values_aa)^2))
rmse_aa
```

```
[1] 167530155600
```

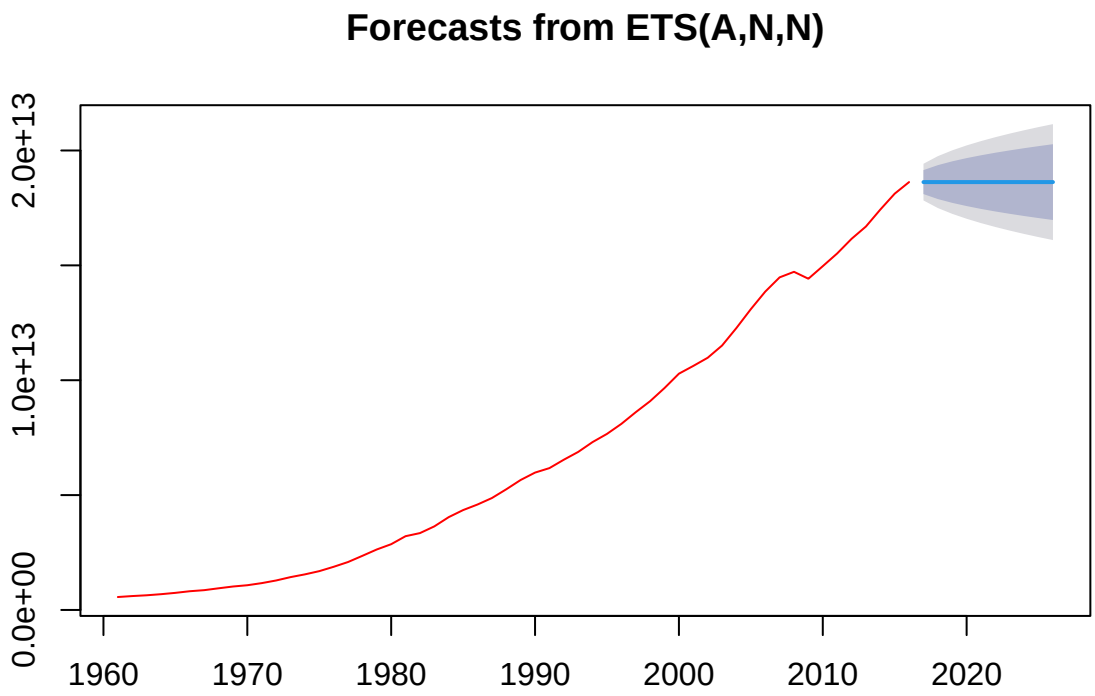
Discuss which model has better fit and whether the more complex model is necessary - Lower RMSE values indicate better model fit because they represent smaller prediction errors. In this case: - The training data had a RSME with 365452825303 and the ETS(A,A,N) model had a RSME of 167530087609 that shows the ETS(A,A,N) mode had a lower and better RSME score. - The RMSE for ETS(A,A,N) is significantly larger than the RMSE for ETS(A,N,N). This suggests that the ETS(A,N,N) model provides a better fit to the training data. - The ETS(A,N,N) model appears to be simpler (with fewer parameters) than the ETS(A,A,N) model and yet performs better in terms of RMSE.

e. Compare the forecasts from both methods. Which do you think is best?

```
plot(forecast(ets_aa_model, h = 10), col = "blue", main = "Comparison of Forecasts")
```



```
plot(forecasted_values, col = "red")
```



Discuss which model you think is best with a brief explanation of why I think the ETS AAN model has the best performance based on the RSME scores and the overall trending patterns in the predictions as shown in the plots from above. - The RMSE for ETS(A,A,N) is significantly larger than the RMSE for ETS(A,N,N). This suggests that the ETS(A,N,N) model provides a better fit to the training data. - The ETS(A,N,N) model appears to be simpler (with fewer parameters) than the ETS(A,A,N) model and yet performs better in terms of RMSE.

Fit ETS so it finds the best fitting model (get fit statistics the fit)

```
best_ets_model <- ets(us_data_ts_tsts)
summary(best_ets_model)
```

```
ETS(M,A,N)

Call:
 ets(y = us_data_ts_tsts)

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.5825
```

```
  Initial states:
    l = 473969866667.206
    b = 72035896650.8411

  sigma:  0.0236

     AIC     AICc      BIC
3069.118 3070.318 3079.245

Training set error measures:
                     ME         RMSE          MAE       MPE     MAPE      MASE
Training set 15465651409 168150083363 104213396865 0.4040544 1.839483 0.3071529
                  ACF1
Training set 0.1288132
```

Use the best fitting model to forecast 10 time points out

```
forecasted_values_best <- forecast(best_ets_model, h = 10)
print(forecasted_values_best)
```

```
     Point Forecast        Lo 80        Hi 80        Lo 95        Hi 95
2017   1.920103e+13 1.862031e+13 1.978176e+13 1.831289e+13 2.008918e+13
2018   1.977757e+13 1.868087e+13 2.087427e+13 1.810032e+13 2.145483e+13
2019   2.035411e+13 1.866351e+13 2.204470e+13 1.776857e+13 2.293965e+13
2020   2.093065e+13 1.856826e+13 2.329303e+13 1.731768e+13 2.454361e+13
2021   2.150718e+13 1.839823e+13 2.461613e+13 1.675246e+13 2.626191e+13
2022   2.208372e+13 1.815629e+13 2.601115e+13 1.607724e+13 2.809020e+13
2023   2.266026e+13 1.784466e+13 2.747585e+13 1.529544e+13 3.002508e+13
2024   2.323679e+13 1.746499e+13 2.900860e+13 1.440958e+13 3.206401e+13
2025   2.381333e+13 1.701845e+13 3.060821e+13 1.342146e+13 3.420520e+13
2026   2.438987e+13 1.650585e+13 3.227389e+13 1.233230e+13 3.644744e+13
```

report the model and interpret each smoothing parameter

```
best_ets_model
```

```
ETS(M,A,N)

Call:
 ets(y = us_data_ts_tsts)

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.5825

  Initial states:
    l = 473969866667.206
    b = 72035896650.8411
```

```
  sigma:  0.0236

     AIC     AICc      BIC
3069.118 3070.318 3079.245
```

Interpret the parameters - Alpha (a): Alpha is the smoothing parameter for the level (l) component. In this model, alpha is estimated to be approximately 0.9999. A value close to 1 suggests that the model relies heavily on recent observations to estimate the current level. - Beta (b): Beta is the smoothing parameter for the trend (b) component. In this model, beta is estimated to be approximately 0.5825. It indicates how much weight is given to the most recent trend when forecasting future values. A value less than 1 suggests that past trends have some influence on the current trend. - Initial States: - Initial Level (l): The estimated initial level is approximately 473,969,866,667.206. It represents the starting point for the level component in the model. - Initial Trend (b): The estimated initial trend is approximately 72,035,905,115.0473. It represents the starting point for the trend component in the model. - Sigma (o): Sigma is the estimate of the error standard deviation, and in this model, it's approximately 0.0236. It measures the variability or volatility of the residuals (errors) in the model. - AIC (Akaike Information Criterion): The AIC value is 3069.118. A lower AIC value suggests that the model fits the data better. It's a measure of model goodness of fit that takes into account model complexity. - AICc (Corrected Akaike Information Criterion): The AICc value is 3070.318. Similar to AIC, but it corrects for small sample sizes. - BIC (Bayesian Information Criterion): The BIC value is 3079.245. Like AIC, it measures model fit while penalizing model complexity. Smaller BIC values indicate better-fitting models. - In summary, this ETS(M,A,N) model has estimated smoothing parameters for level and trend, initial states for level and trend, and a measure of the error standard deviation. The AIC, AICc, and BIC values provide information about the goodness of fit and model complexity. You would typically choose the model with the lowest AIC, AICc, or BIC value, as it indicates the best trade-off between model fit and complexity.

Fit and Forecast Methane Data using ETS

Find and obtain the globally averaged marine surface methane (CH_4) data

Hint: You'll have to copy and paste the data into a .csv and then load

the data into R Here's the website where the data can be found: https://gml.noaa.gov/ccgg/

Prepare Data

```
library(readr)
library(ggplot2)
library(zoo)
library(fabletools)
library(distributional)
methane_data <- read_csv("~/Desktop/data/methane_data.csv")
methane_data
```

```
# A tibble: 466 x 7
    year month decimal average average_unc trend trend_unc
   <dbl> <dbl>   <dbl>   <dbl>       <dbl> <dbl>     <dbl>
```

```
 1  1983     7   1984.   1626.         2.1 1636.       1.4
 2  1983     8   1984.   1628          2.7 1636.       1.3
 3  1983     9   1984.   1638.         2.3 1637.       1.3
 4  1983    10   1984.   1645.         1.6 1637.       1.2
 5  1983    11   1984.   1643.         0.8 1638.       1.1
 6  1983    12   1984.   1640.         1   1638.       1
 7  1984     1   1984.   1639.         1.8 1639.       0.9
 8  1984     2   1984.   1639.         2.1 1640        0.9
 9  1984     3   1984.   1641.         1.8 1641.       0.8
10  1984     4   1984.   1644.         1.9 1642.       0.8
# ... with 456 more rows
```

Set Prediction and Actual Data

- Create an object called `prediction_methane` with the dates from July 1983-July 2017
- Create a separate object called `actual_methane` with the dates from August 2017-April 2022

```
# Convert year and month columns to a Date object
methane_data$Date <- as.Date(paste(methane_data$year, methane_data$month, 1, sep = "-"))

# Define the start and end dates for prediction_methane and actual_methane
start_date_pred <- as.Date("1983-07-01")
end_date_pred <- as.Date("2017-07-01")

start_date_actual <- as.Date("2017-08-01")
end_date_actual <- as.Date("2022-04-01")

# Create the 'prediction_methane' object by filtering the data
prediction_methane <- methane_data[methane_data$Date >= start_date_pred & methane_data$Date <= end_date_

# Create the 'actual_methane' object by filtering the data
actual_methane <- methane_data[methane_data$Date >= start_date_actual & methane_data$Date <= end_date_a

prediction_methane
```

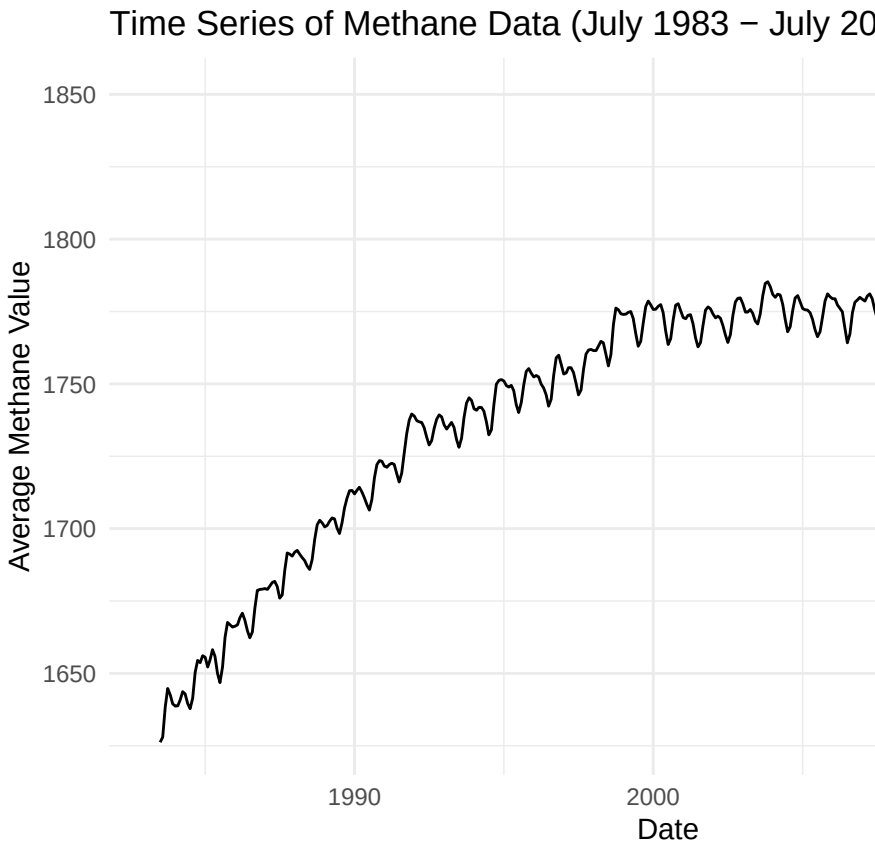
```
# A tibble: 409 x 8
    year month decimal average average_unc trend trend_unc Date
   <dbl> <dbl>   <dbl>   <dbl>       <dbl> <dbl>     <dbl> <date>
 1  1983     7   1984.   1626.         2.1 1636.       1.4 1983-07-01
 2  1983     8   1984.   1628          2.7 1636.       1.3 1983-08-01
 3  1983     9   1984.   1638.         2.3 1637.       1.3 1983-09-01
 4  1983    10   1984.   1645.         1.6 1637.       1.2 1983-10-01
 5  1983    11   1984.   1643.         0.8 1638.       1.1 1983-11-01
 6  1983    12   1984.   1640.         1   1638.       1   1983-12-01
 7  1984     1   1984.   1639.         1.8 1639.       0.9 1984-01-01
 8  1984     2   1984.   1639.         2.1 1640        0.9 1984-02-01
 9  1984     3   1984.   1641.         1.8 1641.       0.8 1984-03-01
10  1984     4   1984.   1644.         1.9 1642.       0.8 1984-04-01
# ... with 399 more rows
```

```
actual_methane
```

```
# A tibble: 57 x 8
    year month decimal average average_unc trend trend_unc Date
   <dbl> <dbl>   <dbl>   <dbl>       <dbl> <dbl>     <dbl> <date>
 1  2017     8   2018.   1845.         1.2 1850.       0.6 2017-08-01
 2  2017     9   2018.   1853.         1   1851.       0.6 2017-09-01
 3  2017    10   2018.   1858.         1   1852.       0.6 2017-10-01
 4  2017    11   2018.   1859.         0.8 1852.       0.7 2017-11-01
 5  2017    12   2018.   1857.         0.9 1853.       0.7 2017-12-01
 6  2018     1   2018.   1854.         1   1854.       0.7 2018-01-01
 7  2018     2   2018.   1855.         1   1854.       0.8 2018-02-01
 8  2018     3   2018.   1857.         1   1855        0.8 2018-03-01
 9  2018     4   2018.   1857.         1.1 1856.       0.8 2018-04-01
10  2018     5   2018.   1855.         1.7 1856.       0.8 2018-05-01
# ... with 47 more rows
```

Plot the Time Series for `prediction_methane`

```
# Create a time series plot for prediction_methane
ggplot(data = prediction_methane, aes(x = Date, y = average)) +
  geom_line() +
  labs(
    title = "Time Series of Methane Data (July 1983 - July 2017)",
    x = "Date",
    y = "Average Methane Value"
  ) +
  theme_minimal()
```



(make sure you label your axes properly)

Fit Best ETS Model and Report Fit for `prediction_methane`

Comment on the smoothing parameters for the model

Discuss what the magnitude of these parameters mean

```r
# Load the forecast library
library(forecast)

# Convert 'Date' to a time series object
prediction_methane_ts <- ts(prediction_methane$average, frequency = 12)

# Fit the best ETS model
best_ets_model <- ets(prediction_methane_ts)

# Print the best ETS model
summary(best_ets_model)
```

Include the type of model

`ETS(A,A,A)`

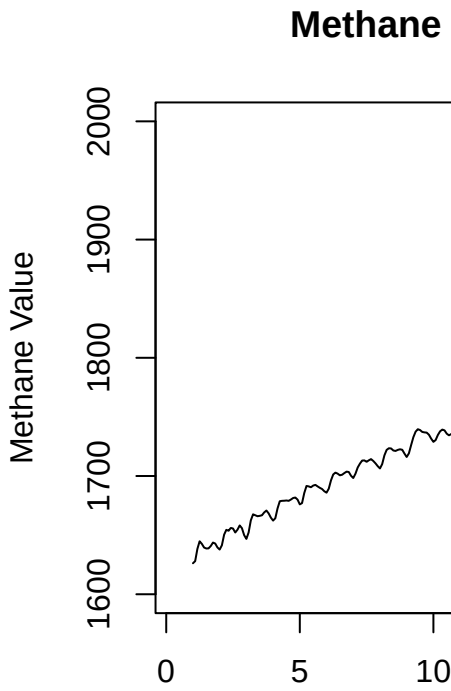
```
Call:
 ets(y = prediction_methane_ts)

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.04
    gamma = 1e-04

  Initial states:
    l = 1632.0715
    b = 1.0839
    s = -5.1957 -0.6992 1.4282 1.5365 1.4122 2.1742
           3.9966 5.3004 4.6701 0.2859 -6.1896 -8.7196

  sigma:  1.1728

     AIC     AICc      BIC
2607.659 2609.224 2675.892

Training set error measures:
                      ME     RMSE       MAE           MPE       MAPE      MASE
Training set -0.02842914 1.149612 0.8786213 -0.001700486 0.05029763 0.1283863
                    ACF1
Training set 0.2780759
```

Discuss what the magnitude of the smoothing parameters mean and what this says about the time series - Model Type: ETS(A, A, A) - Alpha (a): The alpha parameter controls the level smoothing and represents the weight given to the most recent observation when updating the estimated level. A high a (close to 1. 0.9999) indicates that the model is giving a very high weight to the most recent observation when updating the estimated level, making the forecast highly responsive to recent changes. - Beta (b): The beta parameter controls the trend smoothing and represents the weight given to the most recent estimated trend when updating the trend component. b is relatively low at 0.04, suggesting that the trend component is updated with less weight on the most recent trend estimate. - Gamma (y): The gamma parameter is applicable only for models with seasonality (e.g., ETS(A, A, M)). It controls the seasonality smoothing and represents the weight given to the most recent estimated seasonal component when updating the seasonal component. y is very low at 0.0001, indicating that the seasonal component is updated with very little weight on the most recent seasonal estimate, suggesting minimal seasonality in the data. - Sigma represents the estimated standard deviation of the error term, which is 1.1728 in this case. It indicates the variability or volatility of the residuals around the predicted values. - AIC (Akaike Information Criterion): 2607.659 - AICc (Corrected AIC): 2609.224 - BIC (Bayesian Information Criterion): 2675.892 - These information criteria can be used for model selection. Lower values of these criteria generally indicate a better-fitting model. - ME (Mean Error): -0.02842914 - RMSE (Root Mean Squared Error): 1.149612 - MAE (Mean Absolute Error): 0.8786213 - MPE (Mean Percentage Error): -0.001700486 - MAPE (Mean Absolute Percentage Error): 0.05029763 - MASE (Mean Absolute Scaled Error): 0.1283863 - ACF1 (First Autocorrelation): 0.2780759 - These error measures assess the performance of the model on the training data. The RMSE, MAE, MAPE, and MASE are commonly used metrics to evaluate the accuracy of time series forecasts. - In summary, the ETS(A, A, A) model suggests that the time series has an additive error, trend, and seasonal component. The high a value indicates strong responsiveness to recent data for level smoothing, while the low b and y values suggest less responsiveness for trend and seasonality. The model's accuracy can be further assessed using out-of-sample data or cross-validation.

```r
# Creating the sample data
data <- c(1844.6, 1852.8, 1858.1, 1858.7, 1856.7, 1854.5, 1854.9, 1856.8, 1856.7, 1854.8, 1852.0, 1849.
          1851.9, 1860.4, 1865.8, 1866.2, 1866.0, 1865.0, 1865.0, 1866.2, 1865.3, 1861.9, 1858.8, 1858.
          1863.0, 1870.7, 1875.4, 1875.6, 1874.7, 1873.2, 1872.7, 1874.8, 1875.9, 1874.3, 1871.9, 1871.
          1876.5, 1884.7, 1890.1, 1891.9, 1891.8, 1889.5, 1887.6, 1888.7, 1891.2, 1891.6, 1888.5, 1886.
          1892.6, 1902.6, 1908.0, 1909.5, 1909.3, 1908.3, 1908.1, 1909.3, 1909.9)

# Converting to time series with index starting from 31
actual_methane_ts_sub <- ts(data, start=35, frequency=12)
```

Using the best fitting model, make and plot a forecast to April 2022 using `prediction_methane`

```r


# Convert 'Date' to a time series object for 'prediction_methane' and 'actual_methane'
prediction_methane_ts <- ts(prediction_methane$average, frequency = 12)
actual_methane_ts <- ts(actual_methane$average, frequency = 12)

# Fit the best ETS(A, A, A) model
best_ets_model <- ets(prediction_methane_ts)

# Forecast to April 2022
forecast_length <- 12  # Forecast for 12 months (April 2022)
forecast_prediction <- forecast(best_ets_model, h = forecast_length)


# Plot the forecast
plot(forecast_prediction,
     ylim = c(1600, 2000),
     xlab = "Date",
     ylab = "Methane Value",
     main = "Methane Forecast (July 1983 - April 2022)")

# Add the line for 'actual_methane' in the specified color
lines(actual_methane_ts_sub, col = "#D55E00")
```



Plot the line for `actual_methane` over the forecast using the color `"#D55E00"`

Compute point accuracy measures between the forecasted and actual values

```
# Calculate errors
errors <- prediction_methane_ts - actual_methane_ts_sub

# Compute MAE
MAE <- mean(abs(errors))

# Compute MAPE
MAPE <- mean(abs(errors/actual_methane_ts_sub)) * 100

# Compute MSE
MSE <- mean(errors^2)

# Compute RMSE
RMSE <- sqrt(MSE)

# Compute Bias
Bias <- mean(errors)

cat("MAE:", MAE, "\n")
```

Discuss whether the forecast was accurate and whether there was any bias in the forecast

```
MAE: 4.2
```

```
cat("MAPE:", MAPE, "%\n")
```

```
MAPE: 0.2276916 %
```

```
cat("MSE:", MSE, "\n")
```

```
MSE: 17.64
```

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

```
RMSE: 4.2
```

```
cat("Bias:", Bias, "\n")
```

```
Bias: -4.2
```

> Discuss whether the forecast was accurate and whether there was any bias in the forecast - MAE (Mean Absolute Error): 4.2 - The MAE is 4.2, which means that on average, the forecasted values are off by 4.2 units from the actual values. This provides a measure of the magnitude of the forecast errors, but doesn't indicate direction (e.g. whether the forecasted values were typically higher or lower than the actual values). But I think this is a really close prediction. - MAPE (Mean Absolute Percentage Error): 0.2276916% - The MAPE is approximately 0.23%. This indicates that the forecasted values are, on average, within 0.23% of the actual values. This is a very low percentage error, suggesting that the forecast is quite accurate in relative terms, which I think did really well. - MSE (Mean Squared Error): 17.64 - The MSE measures the average squared difference between the forecasted and actual values. A higher MSE indicates larger average squared errors. In isolation, the MSE might not be as intuitive as some of the other metrics. It's particularly sensitive to outliers, meaning a single large error can heavily influence its value. - RMSE (Root Mean Squared Error): 4.2 - The RMSE is the square root of the MSE, giving a measure of the average magnitude of errors in the same units as the original data. It's notable that the RMSE is equal to the MAE in this case, which suggests that the errors are fairly consistent; large errors aren't skewing the results significantly. - Bias: -4.2 - The bias is -4.2, which means that on average, the forecasted values are 4.2 units lower than the actual values. A negative bias indicates a systematic underestimation in the forecasts. This consistent underestimation could be a point of focus for refining the forecasting model or approach. - The forecast appears to be quite accurate based on the MAPE of 0.23%, meaning the relative error is very low.However, there's a consistent underestimation in the forecasts, as indicated by the bias of -4.2. This systematic bias suggests that there might be a structural issue or trend that the forecasting model isn't fully capturing.Both MAE and RMSE being 4.2 (and equal to each other) indicate that there aren't sporadic large errors greatly influencing the forecast accuracy metrics. Errors seem to be fairly consistent. In summary, while the forecast seems accurate in relative terms, there's a clear indication of consistent underestimation.

Compute and compare distribution accuracy measures between the forecasted and actual values

Estimate the best fit model and compare with the ETS(A,A,N) model

Compute forecasts for both models

Compute `winkler_score` and CRPS

```
library(forecast)
library(fabletools)
library(distributional)
library(scoringRules)

best_fit <- ets(prediction_methane_ts)
ets_AAN <- ets(prediction_methane_ts, model="AAN")

forecast_best_fit <- forecast(best_fit)
forecast_AAN <- forecast(ets_AAN)

# Calculate standard deviations
sd_best_fit <- (forecast_best_fit$upper[,1] - forecast_best_fit$mean) / qnorm(0.975)
sd_AAN <- (forecast_AAN$upper[,1] - forecast_AAN$mean) / qnorm(0.975)

# Constructing the approximate distribution for best_fit
dist_best_fit <- dist_normal(mean = forecast_best_fit$mean, sd = sd_best_fit)

# Constructing the approximate distribution for ets_AAN
dist_AAN <- dist_normal(mean = forecast_AAN$mean, sd = sd_AAN)

# Now, using the constructed distributions with winkler_score
winkler_best_fit <- winkler_score(dist_best_fit, actual_methane_ts_sub)
winkler_AAN <- winkler_score(dist_AAN, actual_methane_ts_sub)

# Compute CRPS
crps_best_fit <- CRPS(dist_best_fit, actual_methane_ts_sub)
crps_AAN <- CRPS(dist_AAN, actual_methane_ts_sub)

# Formatting and printing
cat("Winkler Score (Best Fit):", winkler_best_fit, "\n")
```

Discuss which model had the better forecast and whether the additive seasonal component was necessary

```
Winkler Score (Best Fit): 610.5782
```

```
cat("Winkler Score (AAN):", winkler_AAN, "\n")
```

```
Winkler Score (AAN): 575.5331
```

```
cat("CRPS (Best Fit):", crps_best_fit, "\n")
```

```
CRPS (Best Fit): 1.190412
```

```
cat("CRPS (AAN):", crps_AAN, "\n")
```

```
CRPS (AAN): 17.44663
```

Discuss which model had the better forecast and whether the additive seasonal component was necessary - Winkler Score: - The Winkler Score is an interval accuracy measure. A lower score suggests that the forecasted intervals more accurately capture the observed data points. - Best Fit Model: 610.5782 - ETS(AAN) Model: 575.5331 - The ETS(AAN) model has a lower Winkler score, indicating that its forecast intervals were closer to the observed data points than the best fit model.

- CRPS (Continuous Ranked Probability Score): CRPS is another measure to evaluate forecast accuracy, especially for probabilistic forecasts. A lower CRPS indicates a better model in terms of overall predictive performance.
- Best Fit Model: 1.190412
- ETS(AAN) Model: 17.44663
- The Best Fit model has a significantly lower CRPS than the ETS(AAN) model, suggesting better overall predictive accuracy for the best fit model.
- The ETS(AAN) model, which doesn't have an additive seasonal component, provides better interval estimates as per the Winkler Score.
- However, when considering the overall predictive accuracy using CRPS, the Best Fit model (which may contain an additive seasonal component, based on the selection of the 'best fit') is superior.
- It seems there is a trade-off between the interval accuracy and overall predictive performance of the two models. The ETS(AAN) model, without an additive seasonal component, captures the observed data points within its intervals more accurately. However, in terms of the overall distribution of predictions, the Best Fit model performs better.
- Considering both metrics, it might be prudent to examine other aspects of the forecasting process, like the nature of the data, to make a final decision. If the data exhibits strong seasonality, then having an additive seasonal component could be beneficial. If not, the non-seasonal ETS(AAN) model seems to be a reasonable choice for interval predictions. However, if overall predictive distribution performance is a higher priority, the Best Fit model appears to be the better choice.