Assignment 6: Cross-Validation

Peter Ye

2024-04-15

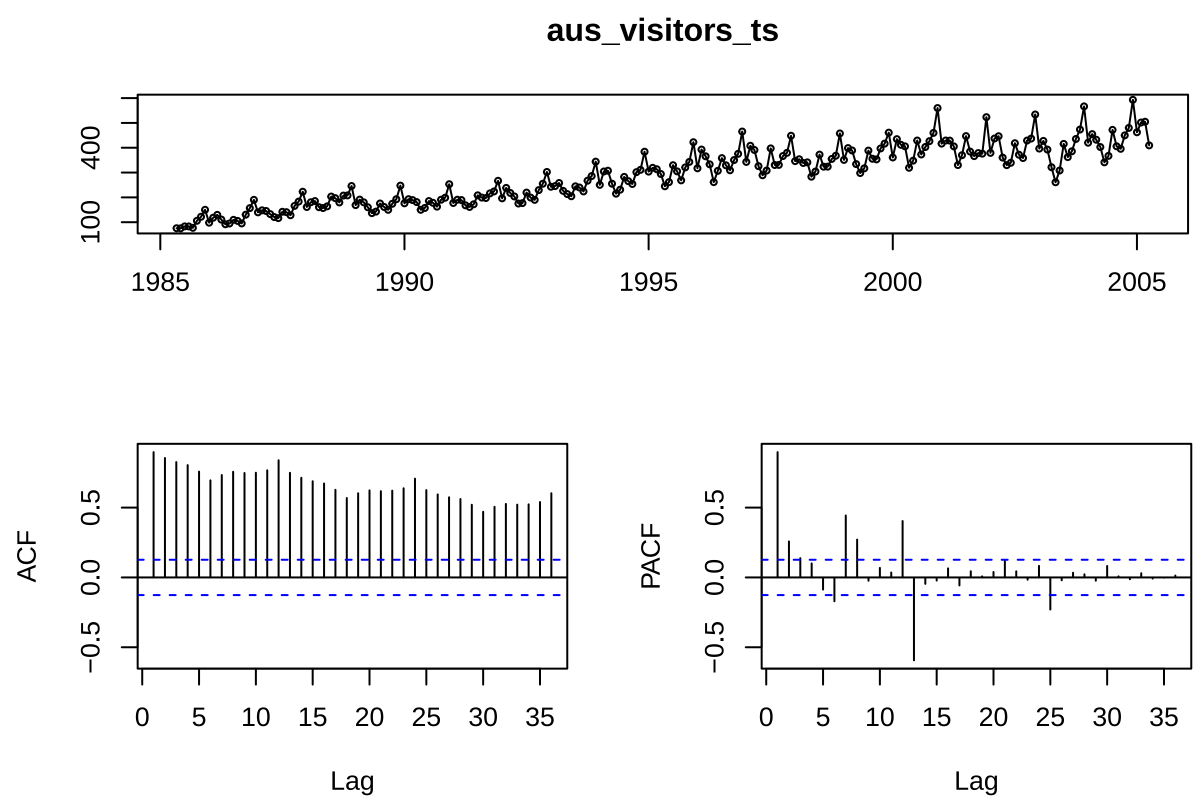
Question 1

```r
# Load data
load("visitors_monthly.rda")
```

```r
# Data preprocessing
aus_visitors <- visitors$x

aus_visitors_ts <- ts(aus_visitors, frequency = 12, start=c(1985, 5), end=c(2005,4))
```

Graph without Box-Cox transformation

```r
# Plot the graph without Box-Cox transformation
tsdisplay(aus_visitors_ts)
```



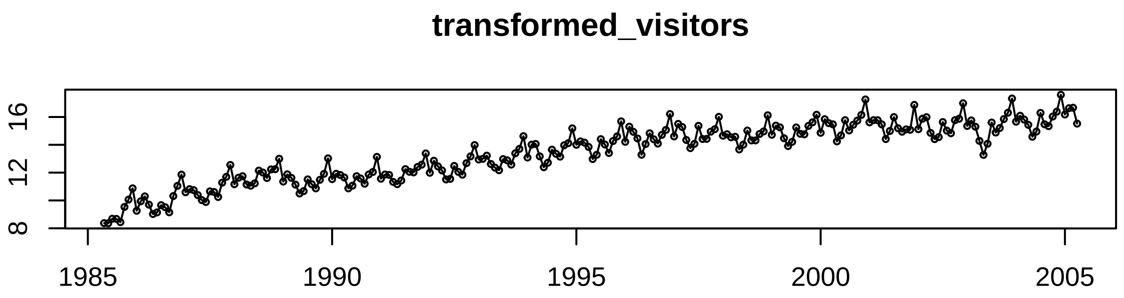
- The time series plot shows fluctuations over time without a clear trend or seasonality.
- The ACF plot shows a slow decay, which could suggest a non-stationary process.
- The PACF plot shows significant spikes at the first lags, indicating potential autoregressive components.

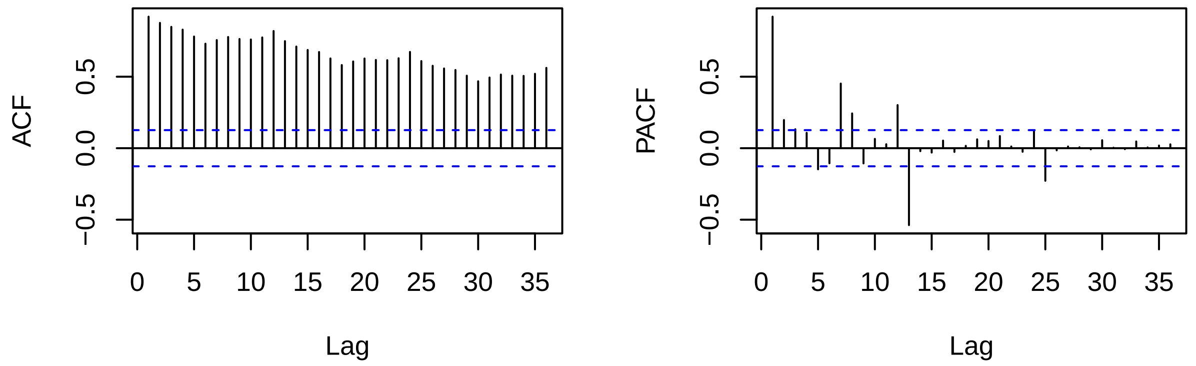
Graph with Box-Cox transformation

```
# Find lambda
lambda <- BoxCox.lambda(aus_visitors_ts)

# Apply the Box-Cox transformation
transformed_visitors <- BoxCox(aus_visitors_ts, lambda)

lambda
```

```
## [1] 0.2775249
```

```
# Plot the graph without Box-Cox transformation
tsdisplay(transformed_visitors)
```





- After transformation, the variance appears more constant over time.
- The ACF plot still shows a slow decay, but perhaps less pronounced than before, suggesting improved but not complete stationarity.
- The PACF plot for the transformed data is similar to the original, suggesting that the transformation has not dramatically changed the autoregressive nature of the data.

Therefore, it appears that implementing a Box-Cox transformation would be beneficial. The purpose of the Box-Cox transformation is to stabilize the variance across the levels of the series, which can be especially helpful when the variance seems to increase with the mean. Such a pattern seems to be present in the visitors data, indicating that a transformation could help in standardizing the variance.

Question 2

Model 1

```
# Create auto arima model
auto_arima_model <- auto.arima(aus_visitors_ts, seasonal=TRUE, stepwise = FALSE, lambda = lambda)
```

```
summary(auto_arima_model)
```

```
## Series: aus_visitors_ts
## ARIMA(3,1,1)(0,1,1)[12]
## Box Cox transformation: lambda= 0.2775249
##
## Coefficients:
##          ar1     ar2      ar3      ma1     sma1
##       0.4625  0.1432  -0.0528  -0.7956  -0.7423
## s.e.  0.1355  0.0849   0.0741   0.1201   0.0451
##
## sigma^2 = 0.06212:  log likelihood = -9.15
## AIC=30.3   AICc=30.68   BIC=50.85
##
## Training set error measures:
##                      ME     RMSE     MAE        MPE     MAPE      MASE
## Training set -0.8376807 15.59557 11.1993 -0.5096885 3.832524 0.4135797
##                     ACF1
## Training set -0.04519124
```

- It indicates an autoregressive part with three lags, a differencing order of 1 to make the series stationary, and a moving average part with one lag
- The coefficients for the ARIMA model are given with their standard errors, and they seem to be significant

Model 2

```r
# Create ets model
ets_model <- ets(aus_visitors_ts, lambda = lambda)
```

```r
summary(ets_model)
```

```
## ETS(A,A,A)
##
## Call:
##  ets(y = aus_visitors_ts, lambda = lambda)
##
##   Box-Cox transformation: lambda= 0.2775
##
##   Smoothing parameters:
##     alpha = 0.613
##     beta  = 1e-04
##     gamma = 0.1629
##
##   Initial states:
##     l = 9
##     b = 0.029
##     s = -0.2608 0.2417 0.2399 -0.1191 1.4225 0.5071
##            0.1414 -0.6578 -0.1895 0.0663 -0.5966 -0.7949
##
##   sigma:  0.2493
##
##      AIC     AICc      BIC
## 665.9353 668.6921 725.1062
##
```

```
## Training set error measures:
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set -0.1254031 15.52155 11.36013 -0.08238026 4.012259 0.4195189
##                     ACF1
## Training set 0.008233928
```

- This model deonstrates that the error, trend, and seasonal components are all additive
- The ARIMA model is more complex in terms of the number of parameters compared to the ETS model, which is more straightforward with its additive components.
- For the ARIMA model, the AIC is significantly lower than that of the ETS model, suggesting that, based on this metric alone, the ARIMA model might have a better fit to the data.

Question 3

```
# Set the parameters
k <- 160
n <- length(aus_visitors_ts)
p <- 12
H <- 12
```

```
# Function to calculate AICc
calculate_aicc <- function(aic, k, n) {
  aicc <- aic + (2 * k * (k + 1)) / (n - k - 1)
  return(aicc)
}
```

```
# Calculate start time
start_time <- tsp(aus_visitors_ts)[1]+(k-2)/p

mae_1 <- matrix(NA,n-k,H)
mae_2 <- matrix(NA,n-k,H)
mae_3 <- matrix(NA,n-k,H)
mae_4 <- matrix(NA,n-k,H)

rmse_1 <- matrix(NA,n-k,H)
rmse_2 <- matrix(NA,n-k,H)
rmse_3 <- matrix(NA,n-k,H)
rmse_4 <- matrix(NA,n-k,H)

aicc_arima_exp <- numeric(n-k)
aicc_arima_slide <- numeric(n-k)
aicc_ets_exp <- numeric(n-k)
aicc_ets_slide <- numeric(n-k)

for(i in 1:(n-k))
{
  # Expanding Window
  expanding_window <- window(aus_visitors_ts, end=start_time + i/p)

  # Sliding Window - keep the training window of fixed length.
  sliding_window <- window(aus_visitors_ts, start=start_time+(i-k+1)/p, end=start_time+i/p)
```

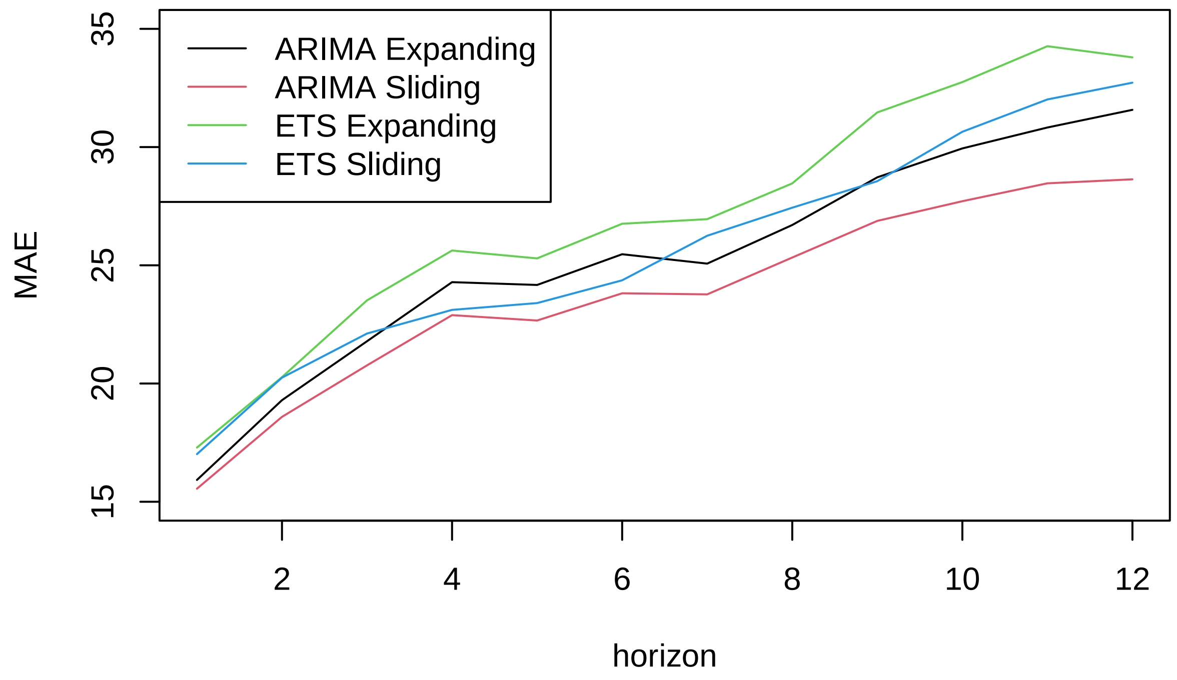
```r
test <- window(aus_visitors_ts, start=start_time + (i+1)/p, end=start_time + (i+H)/p)


if (i<4) {
cat(c("*** CV", i,":","len(Expanding Window):",length(expanding_window), "len(Sliding Window):",lengt
cat(c("*** TRAIN -  Expanding WIndow:",tsp(expanding_window)[1],'-',tsp(expanding_window)[2],'\n'))
cat(c("*** TRAIN - Sliding WIndow:",tsp(sliding_window)[1],'-',tsp(sliding_window)[2],'\n'))
cat(c("*** TEST:",tsp(test)[1],'-',tsp(test)[2],'\n'))
cat("*************************** \n \n")
}

# ARIMA models
arima_expanding <- Arima(expanding_window, order=c(1,0,1), seasonal=list(order=c(0,1,2), period=p),
       include.drift=TRUE, lambda="auto", method="ML")
arima_expanding_fcast <- forecast(arima_expanding, h=H)

arima_sliding <- Arima(sliding_window, order=c(1,0,1), seasonal=list(order=c(0,1,2), period=p),
       include.drift=TRUE, lambda="auto", method="ML")
arima_sliding_fcast <- forecast(arima_sliding, h=H)

# ETS Models
ets_expanding <- ets(expanding_window, model ="MAM")
ets_expanding_fcast <- forecast(ets_expanding, h=H)
ets_sliding <- ets(sliding_window, model ="MAM")
ets_sliding_fcast <- forecast(ets_sliding, h=H)

# Calculate MAE
mae_1[i,1:length(test)] <- abs(arima_expanding_fcast[['mean']]-test)
mae_2[i,1:length(test)] <- abs(arima_sliding_fcast[['mean']]-test)
mae_3[i,1:length(test)] <- abs(ets_expanding_fcast[['mean']]-test)
mae_4[i,1:length(test)] <- abs(ets_sliding_fcast[['mean']]-test)

# Calculate RMSE
rmse_1[i, 1:length(test)] <- sqrt((arima_expanding_fcast[['mean']] - test)^2)
rmse_2[i, 1:length(test)] <- sqrt((arima_sliding_fcast[['mean']] - test)^2)
rmse_3[i, 1:length(test)] <- sqrt((ets_expanding_fcast[['mean']] - test)^2)
rmse_4[i, 1:length(test)] <- sqrt((ets_sliding_fcast[['mean']] - test)^2)

# Calculate AIC values
aic_arima_exp <- AIC(arima_expanding)
aic_arima_slide <- AIC(arima_sliding)
aic_ets_exp <- AIC(ets_expanding)
aic_ets_slide <- AIC(ets_sliding)

# Calculate number of parameters (p + q + 1 for constant if applicable)
k_arima_exp <- length(arima_expanding$coef)
k_arima_slide <- length(arima_sliding$coef)
k_ets_exp <- length(ets_expanding$par)
k_ets_slide <- length(ets_sliding$par)

# Store AICc values
aicc_arima_exp[i] <- calculate_aicc(aic_arima_exp, k_arima_exp, length(expanding_window))
aicc_arima_slide[i] <- calculate_aicc(aic_arima_slide, k_arima_slide, length(sliding_window))
```

```r
  aicc_ets_exp[i] <- calculate_aicc(aic_ets_exp, k_ets_exp, length(expanding_window))
  aicc_ets_slide[i] <- calculate_aicc(aic_ets_slide, k_ets_slide, length(sliding_window))
}
```

```
## *** CV 1 : len(Expanding Window): 160 len(Sliding Window): 160 len(Test): 12
## *** TRAIN -  Expanding WIndow: 1985.33333333333 - 1998.58333333333
## *** TRAIN - Sliding WIndow: 1985.33333333333 - 1998.58333333333
## *** TEST: 1998.66666666667 - 1999.58333333333
## ***************************
##
## *** CV 2 : len(Expanding Window): 161 len(Sliding Window): 160 len(Test): 12
## *** TRAIN -  Expanding WIndow: 1985.33333333333 - 1998.66666666667
## *** TRAIN - Sliding WIndow: 1985.41666666667 - 1998.66666666667
## *** TEST: 1998.75 - 1999.66666666667
## ***************************
##
## *** CV 3 : len(Expanding Window): 162 len(Sliding Window): 160 len(Test): 12
## *** TRAIN -  Expanding WIndow: 1985.33333333333 - 1998.75
## *** TRAIN - Sliding WIndow: 1985.5 - 1998.75
## *** TEST: 1998.83333333333 - 1999.75
## ***************************
##
```

MAE Plot

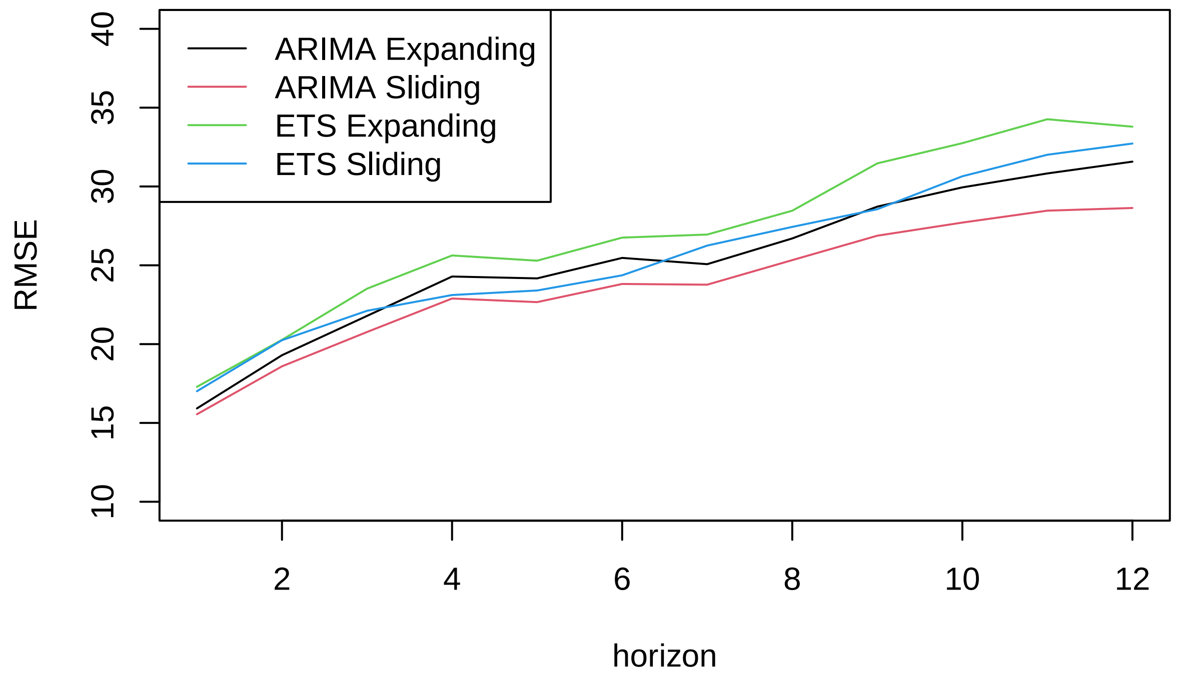
```r
plot(1:12, colMeans(mae_1,na.rm=TRUE), type="l",col=1,xlab="horizon", ylab="MAE",
     ylim=c(15,35))
lines(1:12, colMeans(mae_2,na.rm=TRUE), type="l",col=2)
lines(1:12, colMeans(mae_3,na.rm=TRUE), type="l",col=3)
lines(1:12, colMeans(mae_4,na.rm=TRUE), type="l",col=4)
legend("topleft",legend=c("ARIMA Expanding", "ARIMA Sliding", "ETS Expanding", "ETS Sliding"),col=1:4,l
```



- The ARIMA Sliding model consistently has the lowest MAE across all horizons, suggesting that it performs the best in terms of average forecast accuracy.
- The ETS Expanding model has the highest MAE, indicating it is the least accurate.
- The ARIMA Expanding and ETS Sliding models show very similar performance across the horizons.

RMSE Plot

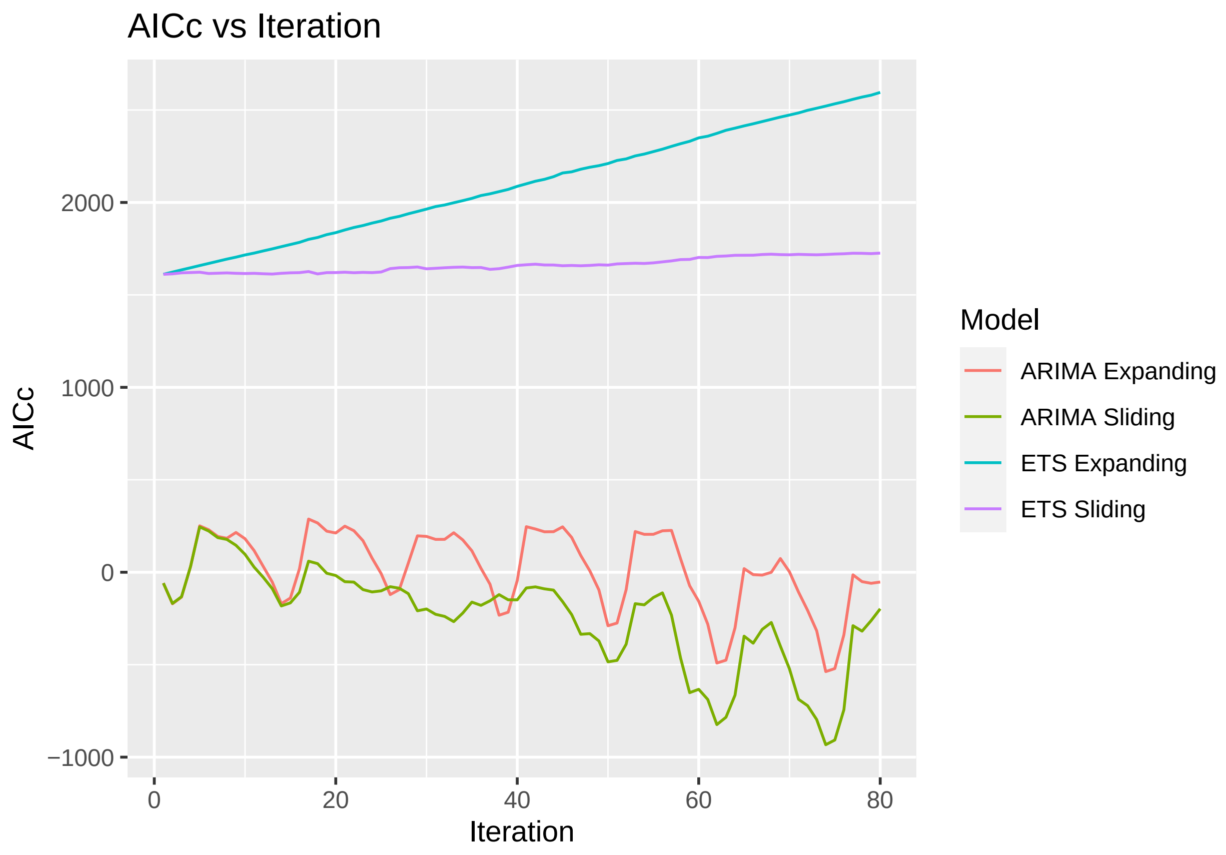
```
# RMSE Plots
plot(1:12, colMeans(rmse_1,na.rm=TRUE), type="l",col=1,xlab="horizon", ylab="RMSE",
     ylim=c(10,40))
lines(1:12, colMeans(rmse_2,na.rm=TRUE), type="l",col=2)
lines(1:12, colMeans(rmse_3,na.rm=TRUE), type="l",col=3)
lines(1:12, colMeans(rmse_4,na.rm=TRUE), type="l",col=4)
legend("topleft",legend=c("ARIMA Expanding", "ARIMA Sliding", "ETS Expanding", "ETS Sliding"),col=1:4,l
```



- The ARIMA Sliding model again performs the best, with the lowest RMSE across all horizons.
- The ETS Expanding model has the highest RMSE, which is consistent with the MAE plot.
- Both the RMSE and MAE plots suggest that the sliding window approach may be superior to the expanding window for the ARIMA model.

AIC Plot

```r
aic_data <- data.frame(
  Iteration = rep(1:(n-k), 4),
  AICc = c(aicc_arima_exp, aicc_arima_slide, aicc_ets_exp, aicc_ets_slide),
  Model = factor(rep(c("ARIMA Expanding", "ARIMA Sliding", "ETS Expanding", "ETS Sliding"), each = n-k)
)
ggplot(aic_data, aes(x = Iteration, y = AICc, colour = Model)) +
  geom_line() +
  labs(title = "AICc vs Iteration", x = "Iteration", y = "AICc")
```



- The ARIMA models, both expanding and sliding, have significantly lower AICc values than the ETS models, suggesting that they have a better fit to the data within the cross-validation.
- There's a fluctuation in the AICc values for ARIMA models, but they tend to stay within a certain range, unlike the ETS models, which have a wider fluctuation.
- Interestingly, the ETS Expanding model's AICc increases linearly, which could indicate an issue with the model as it is applied over more data.

Question 4

In the cross-validation for SARIMA and ETS models, the pre-determined selection of model parameters can be a significant shortcoming. By setting these parameters in advance, based on the entire dataset, there's a risk of 'looking ahead', meaning future information influences the model. This can give a false sense of accuracy because the model is effectively 'cheating' by using knowledge it wouldn't have in a real prediction scenario. Fixed parameters throughout the validation process ignore that real-world data can change, and thus the best model might need to change too. Expanding and sliding window techniques both have pitfalls; the former can give too much weight to old data, and the latter might miss out on important trends present in the full dataset. Complex models like SARIMA are also prone to fitting the noise in the data rather than the signal, especially if there's not enough data for the number of parameters they use.

A more refined approach would harness automated tools like auto.arima and ets to adaptively select the best model parameters during each step of cross-validation. This not only brings in the best aspects of both expanding and sliding windows but also helps the model stay relevant as data evolves.