Forecasting Competition

We've conducted a competition to compare exponential smoothing, an ARMA(10, 10) model and the 'auto.arima' function from the forecast library in R to select an ARMA(p, q) model using AIC.

REDACTED

There is a flaw in the ARMA(10, 10) model. During the competition, it was found that 3 of the 10 datasets could not be used. The prescribed parameters (which loosely correspond to the complexity of the model) are unsuitable.

REDACTED

In other words, this method exists to demonstrate faults.

REDACTED

```
## DATA CLEANING
rm(list=ls())
histdata <- readRDS("projectdata.rds")
datatime <- rep(NA, 10)
for(i in 1:length(histdata)) {
  datatime[i] <- time(histdata[[i]])
}
earliest <- which(datatime == min(datatime))
df <- lapply(histdata, function(X) {
  N <- length(X)
  t <- time(X)
  train_t <- t[1:(N-10)]
  test_t <- t[(N-10 + 1):N]
  test_t <- test_t[1:10]
  list(data = X,
       train = window(X, start=train_t[1], end=train_t[length(train_t)]),
       test = window(X, start=test_t[1], end=test_t[length(test_t)]))
})
```

```
## EXPONENTIAL SMOOTHING
hw <- lapply(1:10, function(X) {
  HoltWinters(df[[X]][[2]], beta = FALSE, gamma = FALSE)
})

hw.predict <- lapply(1:10, function(X) {
  hw.predict <- predict(hw[[X]], n.ahead = 10, prediction.interval = TRUE,
       level = 0.95)
  new.hw <- cbind(hw.predict[,1], time(hw.predict))
  upr.hw <-cbind(hw.predict[,2], time(hw.predict))
  lower.hw <-cbind(hw.predict[,3], time(hw.predict))
  list(hw.predict, new.hw, upr.hw, lower.hw)
})

width <- sapply(1:10, FUN = function(X) {
  (hw.predict[[X]][[3]] - hw.predict[[X]][[4]])[c(1,2,10)]
})

hw.avg.width.1 <- mean(width[1,])
hw.avg.width.2 <- mean(width[2,])
hw.avg.width.10 <- mean(width[3,])

# coverage
hw.cvg.1 <- sum(sapply(1:10, function(X) {
  ifelse((c(hw.predict[[X]][[3]])[1] >= c(df[[X]][[3]])[1] &
```

```
        c(hw.predict[[X]][[4]])[1] <= c(df[[X]][[3]])[1]), 1, 0)
}))/10
hw.cvg.2 <- sum(sapply(1:10, function(X) {
  ifelse((c(hw.predict[[X]][[3]])[2] >= c(df[[X]][[3]])[2] &
        c(hw.predict[[X]][[4]])[2] <= c(df[[X]][[3]])[2]), 1, 0)
}))/10
hw.cvg.10 <- sum(sapply(1:10, function(X) {
  ifelse((c(hw.predict[[X]][[3]])[10] >= c(df[[X]][[3]])[10] &
        c(hw.predict[[X]][[4]])[10] <= c(df[[X]][[3]])[10]), 1, 0)
}))/10

# bias
library(Metrics)

hw.bias.1 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[1], c(hw.predict[[X]][[2]])[1])
}))

hw.bias.2 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[2], c(hw.predict[[X]][[2]])[2])
}))

hw.bias.10 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[10], c(hw.predict[[X]][[2]])[10])
}))
```

```
## ARMA(10, 10)
library(forecast)
arma_predict <- lapply(1:10, function(X) {
  model <- arima(df[[X]][[2]], order=c(10, 0, 10), method = "CSS")
  arma_predict <- forecast(object = model, h = 10, level = 0.95)
  new.arma <- cbind(arma_predict$mean, time(arma_predict$mean))
  lower.arma <- cbind(arma_predict$lower, time(arma_predict$lower))
  upr.arma <-cbind(arma_predict$upper, time(arma_predict$upper))
  list(arma_predict, new.arma, upr.arma, lower.arma)
})

width_arma <- sapply(1:10, FUN = function(X) {
  (arma_predict[[X]][[3]] - arma_predict[[X]][[4]])[c(1,2,10)]
})

arma.avg.width.1 <- mean(width_arma[1,])
arma.avg.width.2 <- mean(width_arma[2,])
arma.avg.width.10 <- mean(width_arma[3,])

# coverage
arma.cvg.1 <- sum(sapply(1:10, function(X) {
  ifelse((c(arma_predict[[X]][[3]])[1] >= c(df[[X]][[3]])[1] &
        c(arma_predict[[X]][[4]])[1] <= c(df[[X]][[3]])[1]), 1, 0)
}))/10
arma.cvg.2 <- sum(sapply(1:10, function(X) {
  ifelse((c(arma_predict[[X]][[3]])[2] >= c(df[[X]][[3]])[2] &
        c(arma_predict[[X]][[4]])[2] <= c(df[[X]][[3]])[2]), 1, 0)
}))/10
arma.cvg.10 <- sum(sapply(1:10, function(X) {
  ifelse((c(arma_predict[[X]][[3]])[10] >= c(df[[X]][[3]])[10] &
        c(arma_predict[[X]][[4]])[10] <= c(df[[X]][[3]])[10]), 1, 0)
}))/10

# bias
library(Metrics)
arma.bias.1 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[1], c(arma_predict[[X]][[2]])[1])
```

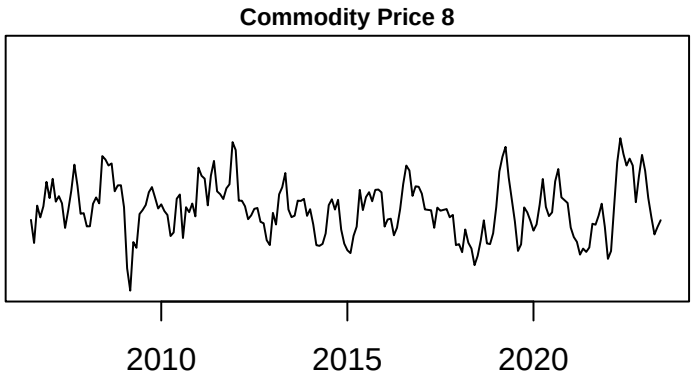
```
}))

arma.bias.2 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[2], c(arma_predict[[X]][[2]])[2])
}))

arma.bias.10 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[10], c(arma_predict[[X]][[2]])[10])
}))
```

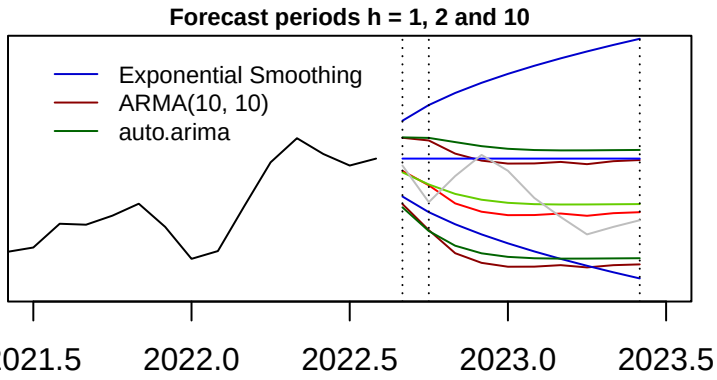
```
auto <- lapply(1:10, function(X) {
  auto.arima(df[[X]][[2]], ic = "aic")
})

auto_predict <- lapply(1:10, function(X) {
  auto_predict <- forecast(object = auto[[X]], h = 10, level = 0.95)
  new.auto <- cbind(auto_predict$mean, time(auto_predict$mean))
  lower.auto <-cbind(auto_predict$lower, time(auto_predict$lower))
  upr.auto <-cbind(auto_predict$upper, time(auto_predict$upper))
  list(arma_predict, new.auto, upr.auto, lower.auto)
})

width_auto <- sapply(1:10, FUN = function(X) {
  (auto_predict[[X]][[3]] - auto_predict[[X]][[4]])[c(1,2,10)]
})

auto.avg.width.1 <- mean(width_auto[1,])
auto.avg.width.2 <- mean(width_auto[2,])
auto.avg.width.10 <- mean(width_auto[3,])

# coverage
auto.cvg.1 <- sum(sapply(1:10, function(X) {
  ifelse((c(auto_predict[[X]][[3]])[1] >= c(df[[X]][[3]])[1] &
        c(auto_predict[[X]][[4]])[1] <= c(df[[X]][[3]])[1]), 1, 0)
}))/10
auto.cvg.2 <- sum(sapply(1:10, function(X) {
  ifelse((c(auto_predict[[X]][[3]])[2] >= c(df[[X]][[3]])[2] &
        c(auto_predict[[X]][[4]])[2] <= c(df[[X]][[3]])[2]), 1, 0)
}))/10
auto.cvg.10 <- sum(sapply(1:10, function(X) {
  ifelse((c(auto_predict[[X]][[3]])[10] >= c(df[[X]][[3]])[10] &
        c(auto_predict[[X]][[4]])[10] <= c(df[[X]][[3]])[10]), 1, 0)
}))/10

# bias
library(Metrics)
auto.bias.1 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[1], c(auto_predict[[X]][[2]])[1])
}))

auto.bias.2 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[2], c(auto_predict[[X]][[2]])[2])
}))

auto.bias.10 <- mean(sapply(1:10, function(X) {
  bias(c(df[[X]][[3]])[10], c(auto_predict[[X]][[2]])[10])
}))
```

REDACTED

The prediction intervals and forecasts for the three methods Exponential Smoothing (blue), ARMA(10, 10) (red) and 'auto.arima' (green) are included as well as the test data (grey) [right plot]:

```r
start <- datatime[8]
par(mfrow=c(1,2), mar = c(2, 1, 1, 1))
plot(df[[8]]$data, xlim=c(start, 2023.5), ylim=c(-5,9), ylab = "",
     yaxt = "n", main = "Commodity Price 8",
     cex.main = 0.75, cex.axis = 0.75)
plot(df[[8]]$train, xlim=c(2021.5, 2023.5), ylim=c(-5,9), ylab = "",
     yaxt = "n", main = "Forecast periods h = 1, 2 and 10",
     cex.main = 0.75, cex.axis = 0.75)
#list(hw.predict, new.hw, upr.hw, lower.hw)

#list(arma_predict, new.arma, upr.arma, lower.arma)

lines(c(time(arma_predict[[8]][[2]][,2])),
      c(arma_predict[[8]][[2]][,1]),
      col = "red")
lines(c(time(arma_predict[[8]][[3]][,2])),
      c(arma_predict[[8]][[3]][,1]),
      col = "darkred")
lines(c(time(arma_predict[[8]][[4]][,2])),
      c(arma_predict[[8]][[4]][,1]),
      col = "darkred")

lines(c(time(hw.predict[[8]][[2]][,2])),
      c(hw.predict[[8]][[2]][,1]),
      col = "blue")
lines(c(time(hw.predict[[8]][[3]][,2])),
      c(hw.predict[[8]][[3]][,1]),
      col = "blue3")
lines(c(time(hw.predict[[8]][[4]][,2])),
      c(hw.predict[[8]][[4]][,1]),
      col = "blue3")

lines(c(time(auto_predict[[8]][[2]][,2])),
      c(auto_predict[[8]][[2]][,1]),
      col = "chartreuse3")
lines(c(time(auto_predict[[8]][[3]][,2])),
      c(auto_predict[[8]][[3]][,1]),
      col = "darkgreen")
lines(c(time(auto_predict[[8]][[4]][,2])),
      c(auto_predict[[8]][[4]][,1]),
      col = "darkgreen")

abline(v=time(hw.predict[[8]][[2]])[1], lty="dotted")
abline(v=time(hw.predict[[8]][[2]])[2], lty="dotted")
abline(v=time(hw.predict[[8]][[2]])[10], lty="dotted")

test <- df[[8]]$test
lines(test, col="grey")

legend(2021.5, 9, legend=c("Exponential Smoothing", "ARMA(10, 10)", "auto.arima"),
       col=c("blue3", "darkred", "darkgreen"),
       lty = c(1, 1, 1), cex=0.75, box.lty = 0)
```





REDACTED

```
library(knitr)
width.hw <- c("Exponential Smoothing", round(hw.avg.width.1, 3), round(hw.avg.width.2, 3), round(hw.avg.width.10,
width.arma <- c("ARMA(10, 10)", round(arma.avg.width.1, 3), round(arma.avg.width.2, 3), round(arma.avg.width.10,
width.auto <- c("Auto ARIMA", round(auto.avg.width.1, 3), round(auto.avg.width.2, 3), round(auto.avg.width.10, 3)

width <- cbind(width.hw, width.arma, width.auto)
rownames(width) <- c("", "1", "2", "10")
colnames(width) <- c("", "Avg. P.I. Width", "")

# kable(width, digits = 3, align = "c", caption =
#           "Average 95%-Prediction Interval Width", booktabs = TRUE)
```

```
bias.hw <- c("Exponential Smoothing", round(hw.bias.1, 3), round(hw.bias.2, 3), round(hw.bias.10, 3))
bias.arma <- c("ARMA(10, 10)", round(arma.bias.1, 3), round(arma.bias.2, 3), round(arma.bias.10, 3))
bias.auto <- c("Auto ARIMA", round(auto.bias.1, 3), round(auto.bias.2, 3), round(auto.bias.10, 3))

bias <- cbind(bias.hw, bias.arma, bias.auto)
rownames(bias) <- c("", "1", "2", "10")
colnames(bias) <- c("", "Avg. Bias", "")

#kable(bias, digits = 3, align = "c", caption = "Average Bias", booktabs = TRUE)
```

```
library(xtable)
cvg.hw <- c("Exponential Smoothing", round(hw.cvg.1, 3), round(hw.cvg.2, 3), round(hw.cvg.10, 3))
cvg.arma <- c("ARMA (10, 10)", round(arma.cvg.1, 3), round(arma.cvg.2, 3), round(arma.cvg.10, 3))
cvg.auto <- c("Auto ARIMA", round(auto.cvg.1, 3), round(auto.cvg.2, 3), round(auto.cvg.10, 3))

cvg <- cbind(cvg.hw, cvg.arma, cvg.auto)
rownames(cvg) <- c("", "1", "2", "10")
colnames(cvg) <- c("", "Coverage", "")
```

```
kable(cbind(width, bias, cvg), digits = 3, caption = NULL,
      booktabs = TRUE, align = "c")
```

	Avg. P.I. Width			Avg. Bias			Coverage		
	Exponential Smoothing	ARMA(10, 10)	Auto ARIMA	Exponential Smoothing	ARMA(10, 10)	Auto ARIMA	Exponential Smoothing	ARMA (10, 10)	Auto ARIMA
1	4.122	3.725	4.014	-0.166	-0.125	-0.067	0.9	0.9	0.9
2	5.551	4.826	5.379	-0.159	-0.115	0.054	0.9	0.9	0.9
10	11.875	6.793	9.33	-0.263	0.561	0.251	1	1	1

REDACTED

We conclude that the models chosen by 'auto.arima' are most suitable.