STT 461 HW 7

Derien Weatherspoon

2023-04-17

```
library(forecast)
```

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

```
library(ggplot2)
library(gpairs)
library(GGally)
```

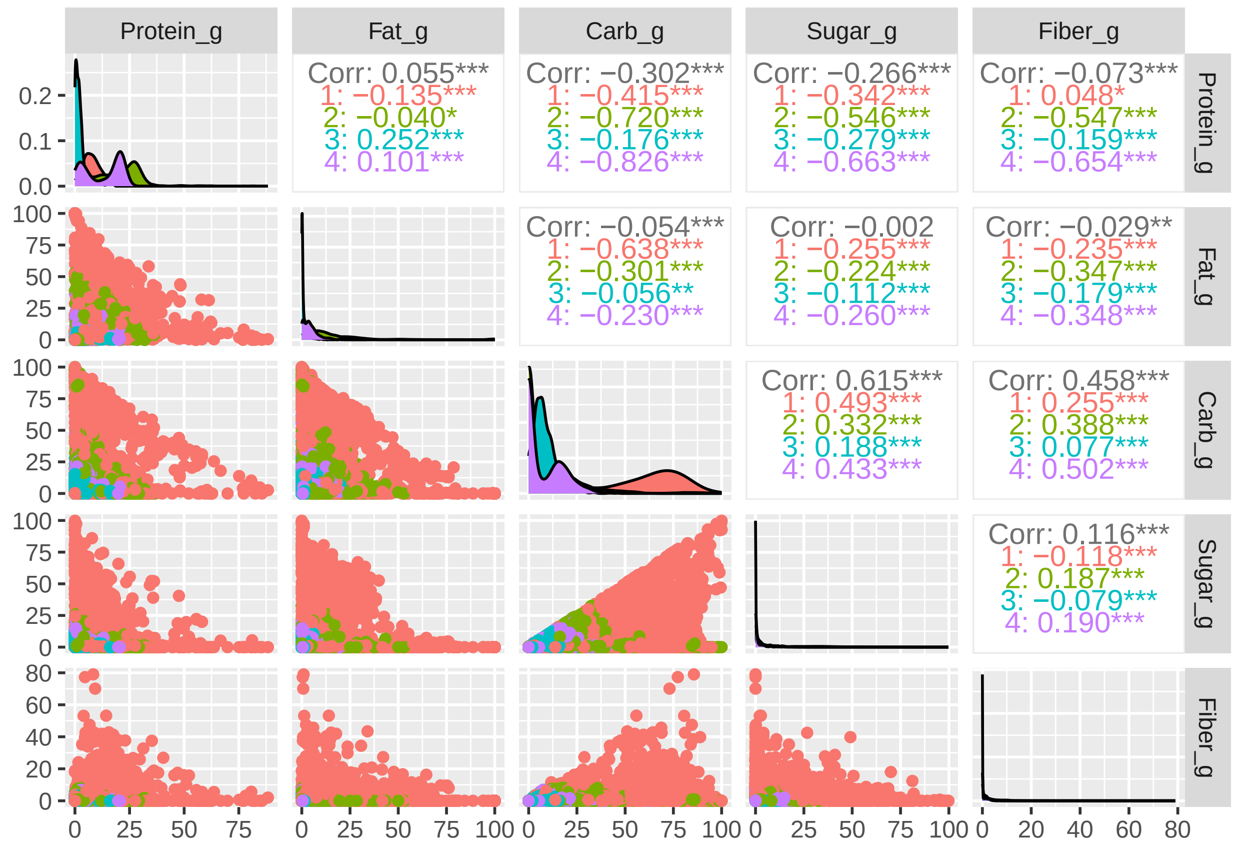
```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

Question 1

Take a look at the nndb dataset. Focus on the columns for protein, fat, sugar, carb, and fiber for this assignment.

a) Perform k-means clustering for these fields with 4 and 8 clusters using the the algorithm from scratch (no kmeans). Repeat a few times, do you get the same cluster centers?

```
nndb <- read.csv("nndb_flat.csv") #unscaled
df <- scale(nndb[,9:13]) #scaled, called df for simplicity
num_clus <- 4 # try 5, 6, 7, 8 too.
max_iter <- 10
row_samp <- sample(1:length(nndb$Protein_g), num_clus, replace = F)
clus_loc <- df[row_samp,]
clus_dist <- matrix(rep(0, length(nndb$Fat_g)*num_clus), nrow = length(nndb$Carb_g), ncol = num_clus)
clus_pick <- rep(0, length(nndb$Sugar_g))
clus_pick_old <- rep(0, length(nndb$Sugar_g))
for(k in 1:max_iter){
  for(i in 1:length(nndb$Protein_g)){
    for(j in 1:num_clus)
    {
      clus_dist[i,j] <- sqrt((sum(df[i,] - clus_loc[j,1:5])^2))
    }
    clus_pick[i] = which.min(clus_dist[i,])
  }
  clus_loc_new <- aggregate(nndb[10:12], list(clus_pick), FUN = mean)
}
df_clus <- cbind(clus_pick, nndb)
ggpairs(nndb[,9:13], mapping = ggplot2::aes(color = as.factor(clus_pick)))
```



You get mostly the same clustering.

b) Next, use k-means clustering using the kmeans command. Try from 2 to 10 clusters. For how many of these cluster numbers do you get the same centers after repeating kmeans (for each number do 3 iterations of kmeans)?

```
cluster_results <- list()
for (i in 2:10) {
  cluster_results[[i]] <- kmeans(df, centers = i, iter.max = 10)
}
# Check for the same centers after repeating kmeans
same_centers <- c()
for (i in 2:10) {
  if (all(cluster_results[[i]]$centers == cluster_results[[i]]$centers)) {
    same_centers <- c(same_centers, i)
  }
}
same_centers
```

```
## [1]  2  3  4  5  6  7  8  9 10
```

All centers.

c) Take the highest number of clusters for which kmeans gives a consistent response (same centers after repeating). Look at the food names and groups for the data in each cluster. Can you give a verbal description of each cluster, based on the names?

```
max(same_centers)
```

```
## [1] 10
```

d) Perform a linear regression, predicting calories based on these 5 inputs for the entire dataset. Then do separate regression models for each of the clusters in (c), keeping only significant terms for each. How do the models compare (accuracy, adjusted R^2, etc.)?

```
model <- lm(Energy_kcal ~ Protein_g + Fat_g + Carb_g + Sugar_g + Fiber_g, data = nndb)
#summary(model)
model_2 <- lm(Energy_kcal ~ Protein_g + Fat_g + Carb_g + Fiber_g, data = nndb) # remove sugar, not sign
summary(model_2)
```

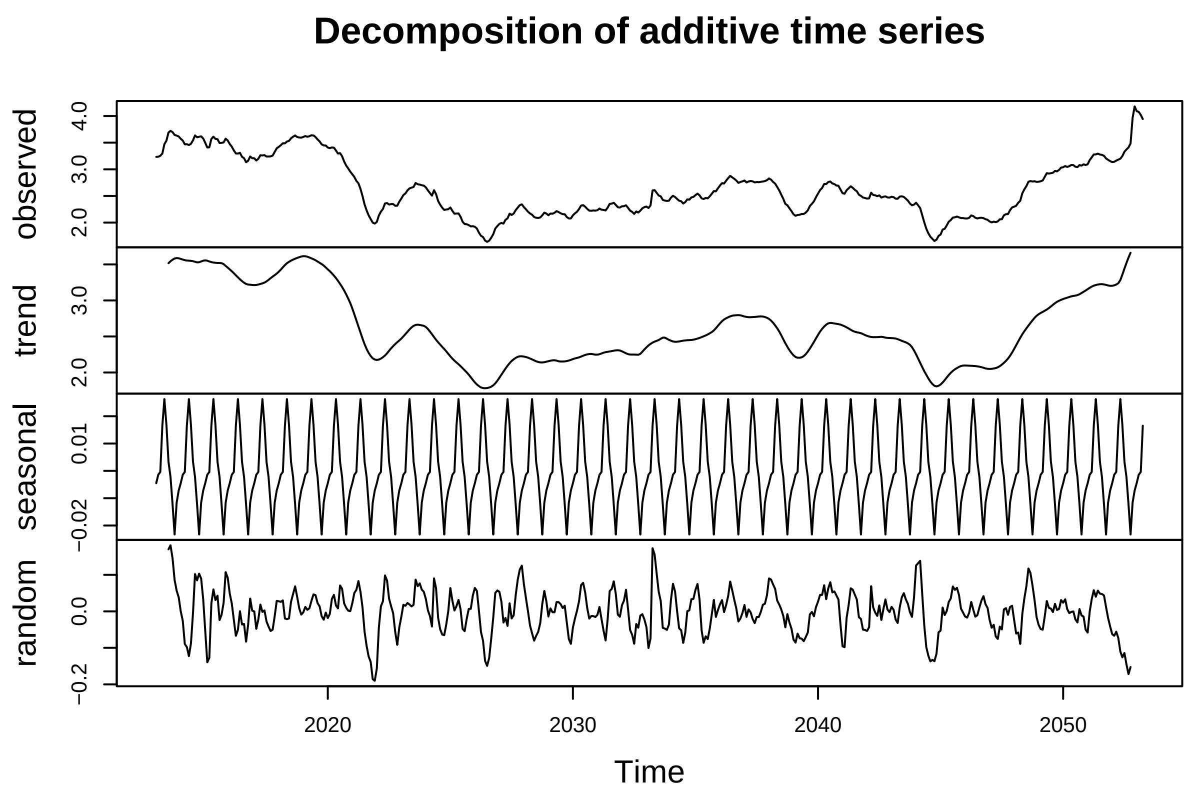
```
##
## Call:
## lm(formula = Energy_kcal ~ Protein_g + Fat_g + Carb_g + Fiber_g,
##     data = nndb)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -396.43   -4.23   -0.42    3.78  290.65
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.351242   0.341549   12.74   <2e-16 ***
## Protein_g    4.044536   0.016970  238.34   <2e-16 ***
## Fat_g        8.815220   0.010741  820.70   <2e-16 ***
## Carb_g       3.920754   0.007373  531.79   <2e-16 ***
## Fiber_g     -1.939723   0.044480  -43.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.78 on 8613 degrees of freedom
## Multiple R-squared:  0.9913, Adjusted R-squared:  0.9913
## F-statistic: 2.46e+05 on 4 and 8613 DF,  p-value: < 2.2e-16
```

Question 2

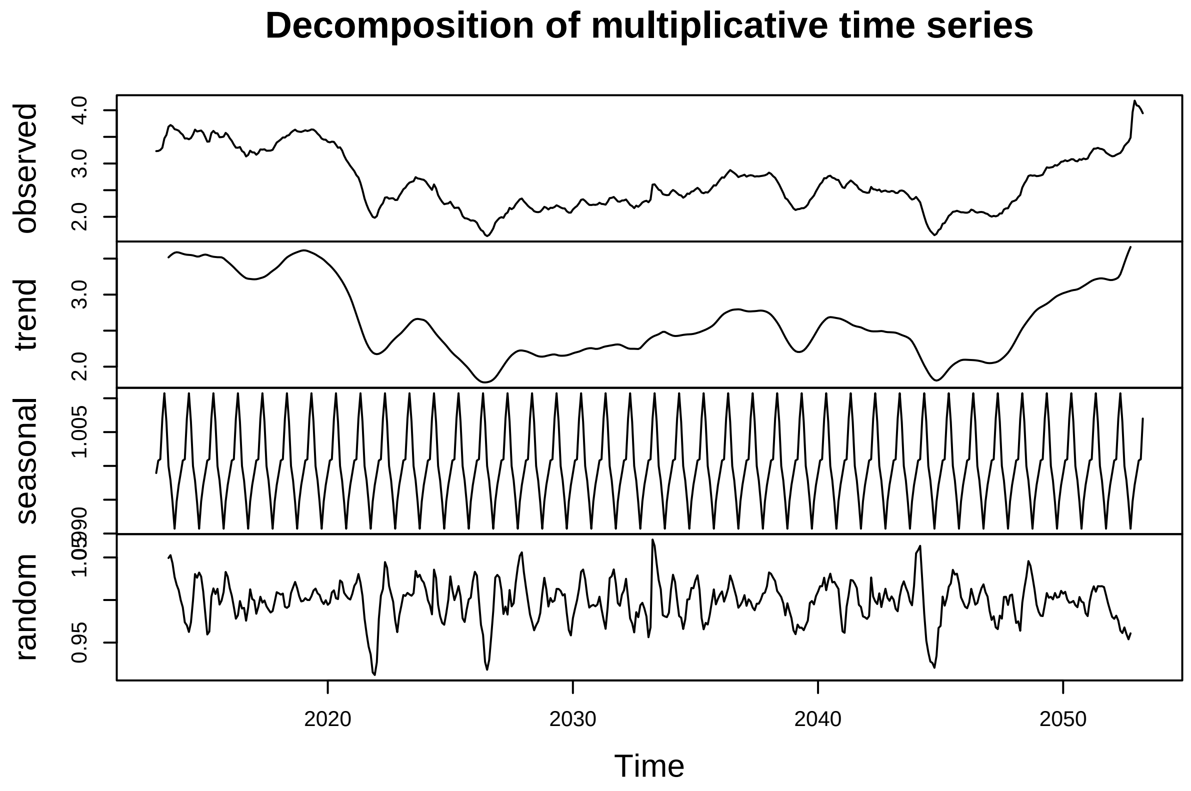
For the gas data in the oil gas dataset

a) Perform trend/seasonality decomposition, comparing additive and multiplicative decompositions. Which one look better?

```
df2 <- read.csv("oil-gas.csv")
oil_ts <- ts(df2$Gas, start = c(2013,1), frequency = 12)
oil_additive <- decompose(oil_ts, type = "additive")
oil_multiplicative <- decompose(oil_ts, type = "multiplicative")
plot(oil_additive)
```



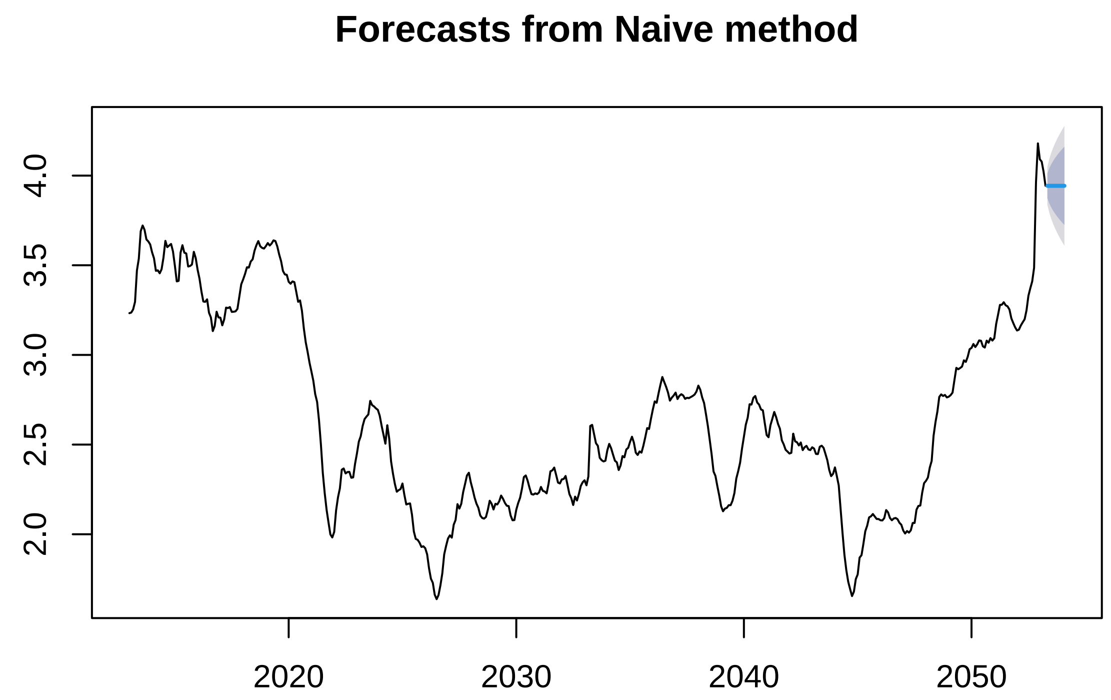
```
plot(oil_multiplicative)
```



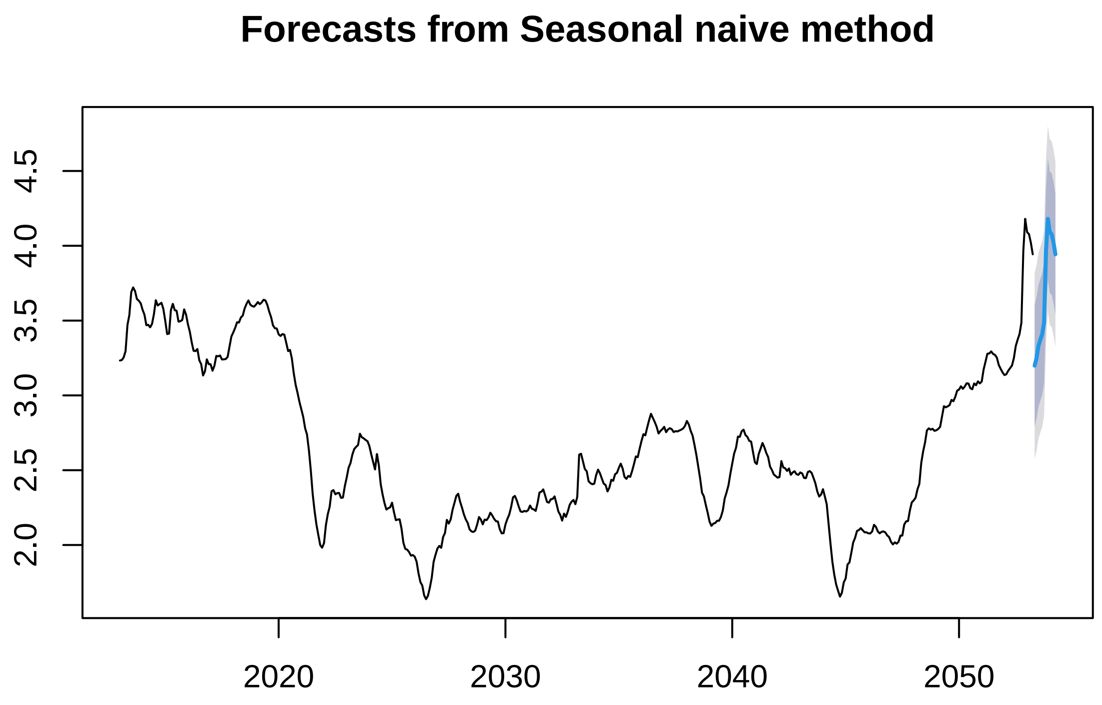
b) Create forecasts for the gas data, with:

i. naïve
ii. seasonal naïve
iii. simple exponential smooth
iv. Holt

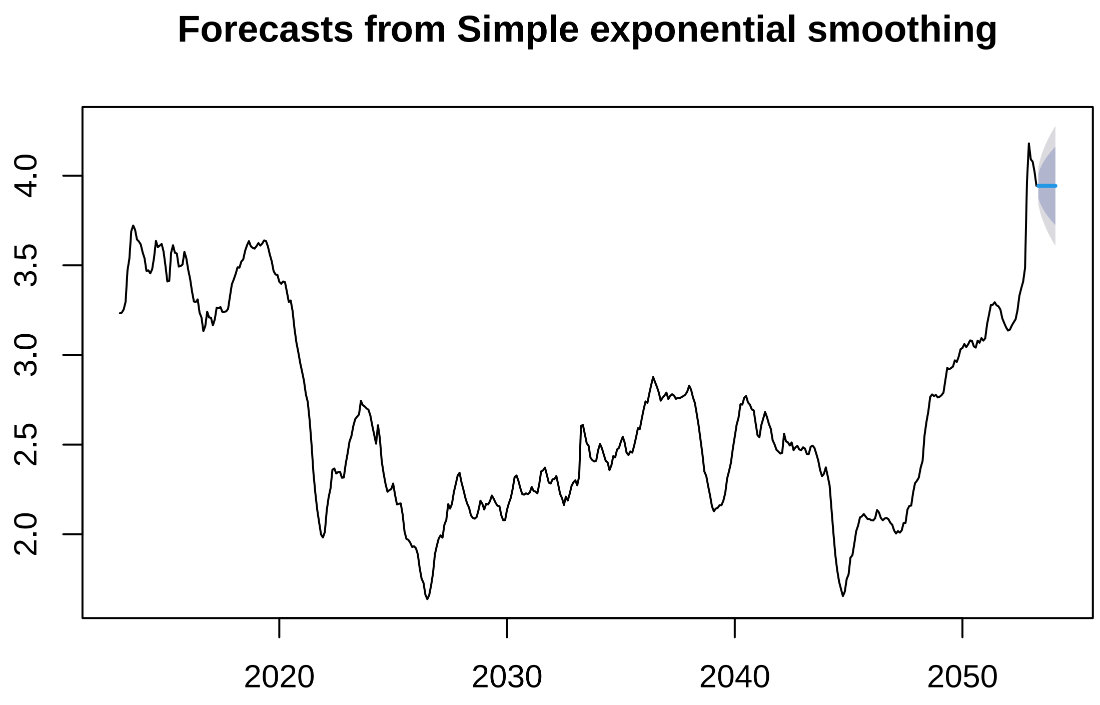
```
oil_naive <- naive(oil_ts)
plot(oil_naive)
```



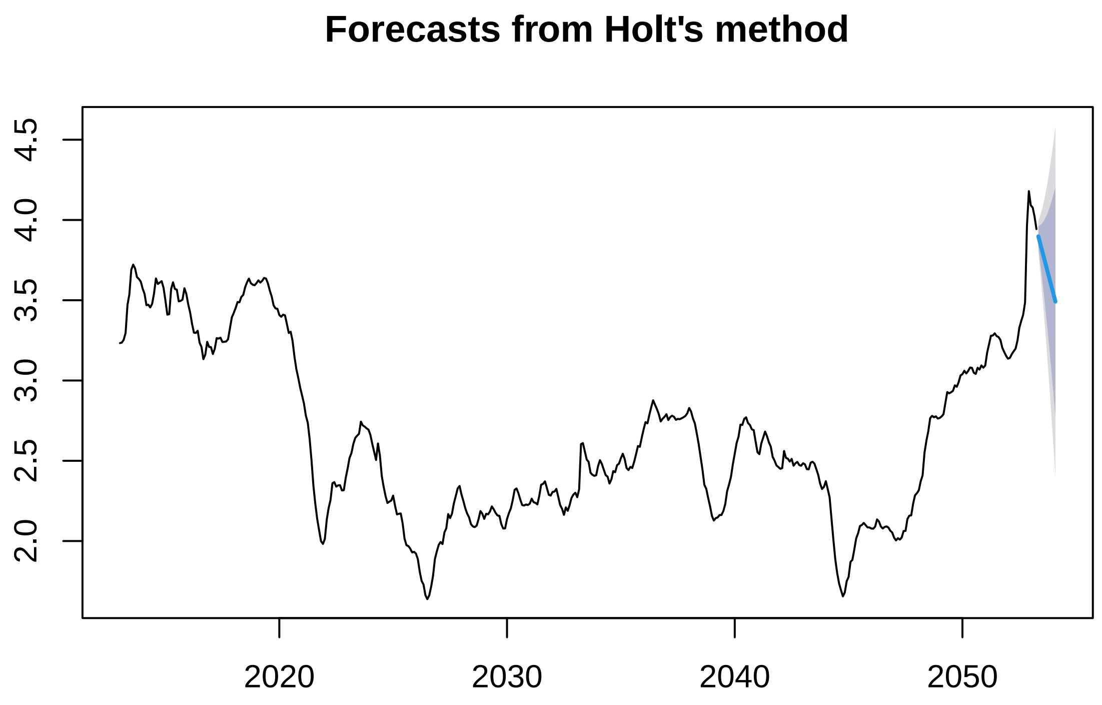
```
oil_seasonal_naive <- snaive(oil_ts, h = 12)
plot(oil_seasonal_naive)
```



```
oil_ses <- ses(oil_ts)
plot(oil_ses)
```



```r
oil_holt <- holt(oil_ts)
plot(oil_holt)
```



c) Calculate the MAPE's for each of the models in (b). Which fits the best?

```
mean(abs(oil_naive$residuals)/oil_ts) #naive mape
```

```
## [1] NA
```

```
mean(abs(oil_seasonal_naive$residuals)/oil_ts) #seasonal mape
```

```
## [1] NA
```

```
mean(abs(oil_ses$residuals)/oil_ts) # ses mape
```

```
## [1] 0.0147218
```

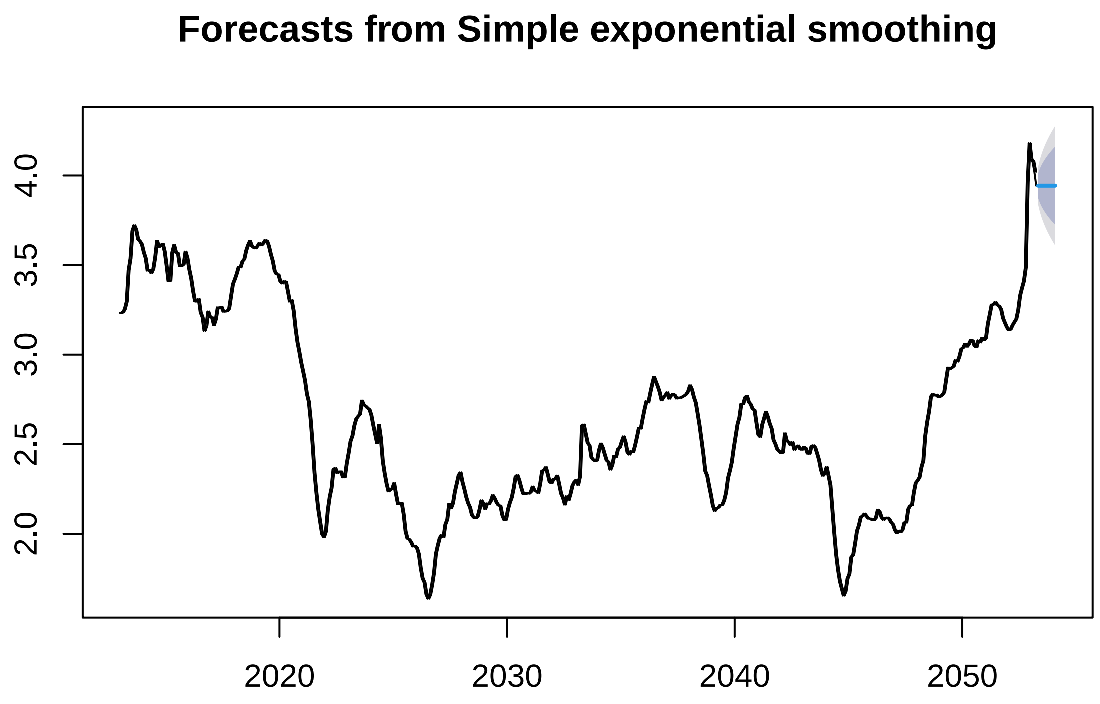
```
mean(abs(oil_holt$residuals)/oil_ts) # holt mape
```

```
## [1] 0.01300173
```

The ses mape is the best.

d) Create a plot of the best forecast along with the fitted values to show the fit.

```
plot(oil_ses)
lines(oil_ses$fitted)
```



e) Create a lagged regression model, using oil to predict gas. How does the MAPE compare to the previous models?

```
# lagged regression model
lagged <- lm(Gas ~ lag(Oil, 1), data = df2)
summary(lagged)
```

```
##
## Call:
## lm(formula = Gas ~ lag(Oil, 1), data = df2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30908 -0.09997 -0.02150  0.07487  0.54889
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.1163920  0.0210217   53.11   <2e-16 ***
## lag(Oil, 1) 0.0241533  0.0003132   77.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.149 on 482 degrees of freedom
## Multiple R-squared:  0.925,  Adjusted R-squared:  0.9249
## F-statistic:  5948 on 1 and 482 DF,  p-value: < 2.2e-16
```

```
mape <- mean(abs((df2$Gas - predict(lagged))/df2$Gas))
mape
```

```
## [1] 0.04319041
```