ENTREGA SERIES TEMPORALES

UXÍA TAMBOADA NIETO

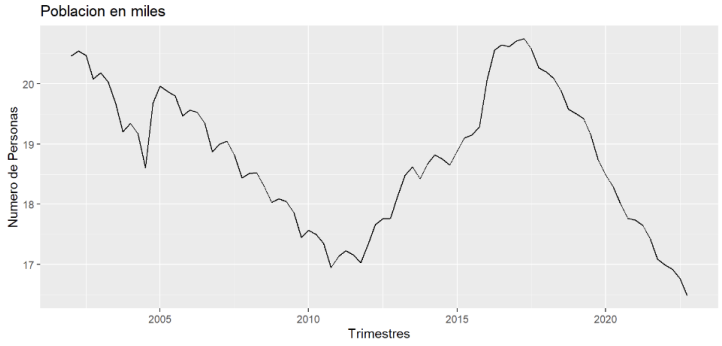
MASTER BD & BUSINESS ANALYTICS

23/02/2023

1. Los datos se han obtenido del INE, se corresponden con los datos de población ocupada en España desde el año 2022 hasta el 2022. La frecuencia es por trimestres, es decir que en cada año tenemos 4 valores. El objetivo de este trabajo es analizar esta serie estacional y obtener un modelo predictivo de la tendencia de la serie en un futuro posterior a los valores de nuestra serie y analizar los resultados obtenidos.

2. Representación gráfica y descomposición de la misma.

```
poblacion <- read_delim("Tarea Series Temporales/poblacion.csv",
                        delim = ";", escape_double = FALSE, trim_ws = TRUE)

Poblacion <- ts(poblacion[,-1], start=c(2002,1), frequency=4)

Poblacion

autoplot(Poblacion)+  ggtitle("Poblacion en miles") +
  xlab("Trimestres") +  ylab("Numero de Personas")
```



La serie no es estacionaria porque ni la media ni la variabilidad son constantes.

Es una serie con cierta tendencia (no es clara) pareciera que empieza en un máximo, luego desciende y vuelve ascender a partir de 2011 para llegar a otro máximo sobre el 2018 y volver a descender.

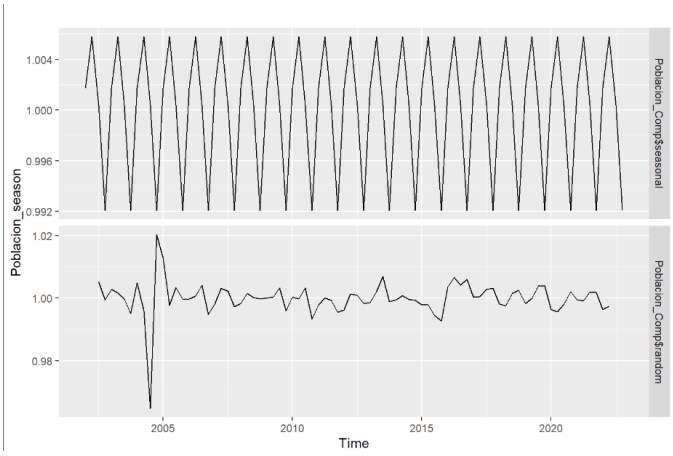
```
#2. Descomposición de la serie temporal en sus componentes.

Poblacion_Comp<- decompose(Poblacion,type=c("multiplicative"))

#Se muestran los coeficientes de estacionalidad

print(Poblacion_Comp$seasonal)
```

	Qtr1	Qtr2	Qtr3	Qtr4
2002	1.0017127	1.0058110	1.0004074	0.9920689
2003	1.0017127	1.0058110	1.0004074	0.9920689
2004	1.0017127	1.0058110	1.0004074	0.9920689
2005	1.0017127	1.0058110	1.0004074	0.9920689
2006	1.0017127	1.0058110	1.0004074	0.9920689
2007	1.0017127	1.0058110	1.0004074	0.9920689
2008	1.0017127	1.0058110	1.0004074	0.9920689
2009	1.0017127	1.0058110	1.0004074	0.9920689
2010	1.0017127	1.0058110	1.0004074	0.9920689
2011	1.0017127	1.0058110	1.0004074	0.9920689
2012	1.0017127	1.0058110	1.0004074	0.9920689
2013	1.0017127	1.0058110	1.0004074	0.9920689
2014	1.0017127	1.0058110	1.0004074	0.9920689
2015	1.0017127	1.0058110	1.0004074	0.9920689
2016	1.0017127	1.0058110	1.0004074	0.9920689
2017	1.0017127	1.0058110	1.0004074	0.9920689
2018	1.0017127	1.0058110	1.0004074	0.9920689
2019	1.0017127	1.0058110	1.0004074	0.9920689
2020	1.0017127	1.0058110	1.0004074	0.9920689
2021	1.0017127	1.0058110	1.0004074	0.9920689
2022	1.0017127	1.0058110	1.0004074	0.9920689

Vemos que para cada trimestre la serie presenta diferentes estacionalidades pero que se repiten para todos los años. El trimestre que presenta mayor estacionalidad es el segundo y el que menor el cuarto trimestre. En particular esto significa que en el segundo trimestre del año se incrementa un 10% más la población activa con respecto a la media de ocupación anual, es decir que hay más nacimientos el segundo trimestre del año y viceversa, el último trimestre del año es el que menos población ocupada presenta.

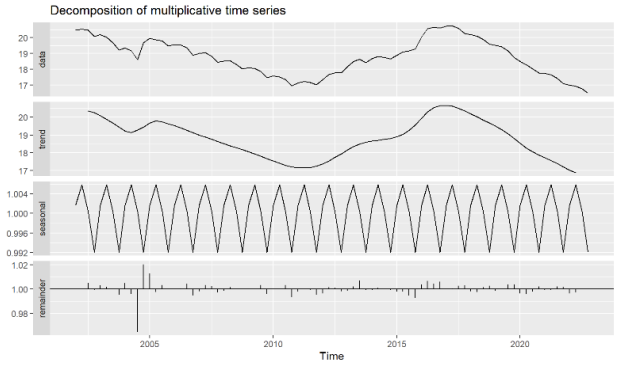
#Representar gráficamente los coeficientes de estacionalidad y la irregularidad

```
Poblacion_season<- cbind(Poblacion_Comp$seasonal,Poblacion_Comp$random)

autoplot(Poblacion_season,facets=TRUE)
```



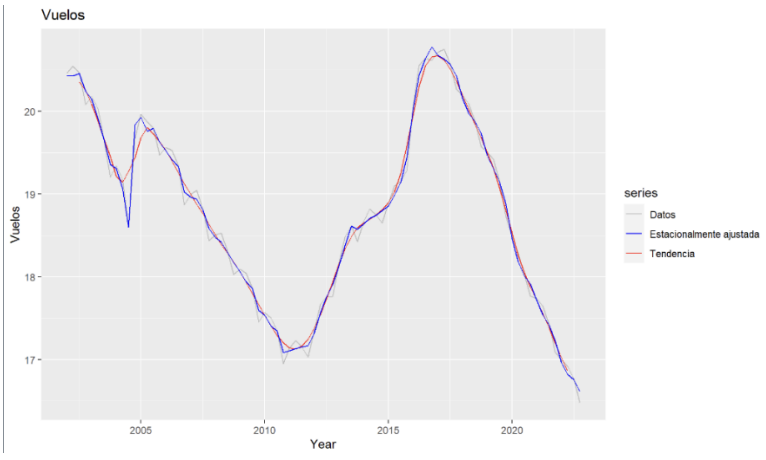
#Representar la serie, la componente estacional, la estimación de la tendencia y el error

```
autoplot(Poblacion_Comp)
```



Como vemos analizando la tendencia, en los últimos años se está produciendo una reducción de la población activa considerable, al igual que ocurrió a partir del 2005 hasta el 2012 aproximadamente. Observando la serie de estacionalidad, vemos que el comportamiento de la serie se repite anualmente aproximadamente.

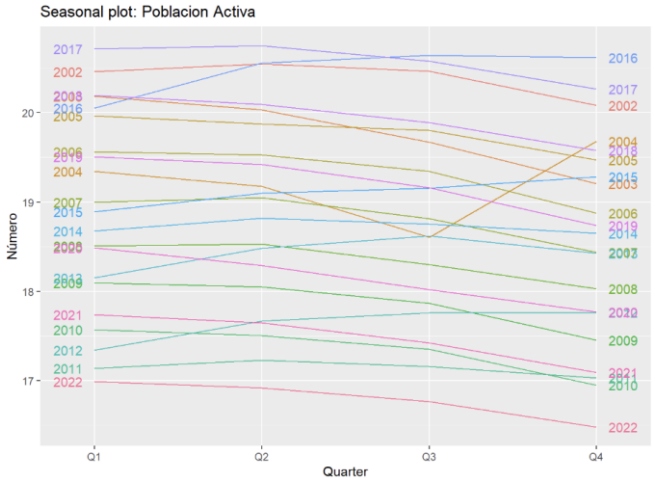
```
#Representar sobre la serie de vuelos original, la tendencia

#calculada con la descomposición y la serie ajustada estacionalmente

autoplot(Poblacion, series="Datos") +

    autolayer(trendcycle(Poblacion_Comp), series="Tendencia") +

    autolayer(seasadj(Poblacion_Comp), series="Estacionalmente ajustada") +

    xlab("Year") + ylab("Vuelos") +

    ggtitle("Vuelos") +

    scale_colour_manual(values=c("gray","blue","red"),

                        breaks=c("Datos","Estacionalmente ajustada","Tendencia"))
```



En azul se representa la serie desestacionalizada en la que no se representa el patrón estacional. Se desestacionaliza dividiendo la serie original entre el coeficiente de estacionalidad que le corresponde para cada valor.

```
#Representar las series de cada año
```

```
ggseasonplot(Poblacion, year.labels=TRUE, year.labels.left=TRUE) +
    ylab("Número") +
    ggtitle("Seasonal plot: Poblacion Activa")
```



El año con más población activa ha sido el 2017 en el segundo trimestre del año y el que menos 2022 el último trimestre.

3. Eficacia del modelo

Antes de empezar a predecir, nos reservamos unos pocos datos para ver la eficacia del modelo predictivo. Como la serie es estacional, nos reservamos un periodo de la misma, que en este caso serán los valores del cuarto trimestre del 2022, que serán los datos que usaremos para comprobar la eficacia del modelo. Para acortar la serie usamos la siguiente función.

```
Poblacion_TR<-window(Poblacion,start=c(2002,1), end=c(2022,4))
```

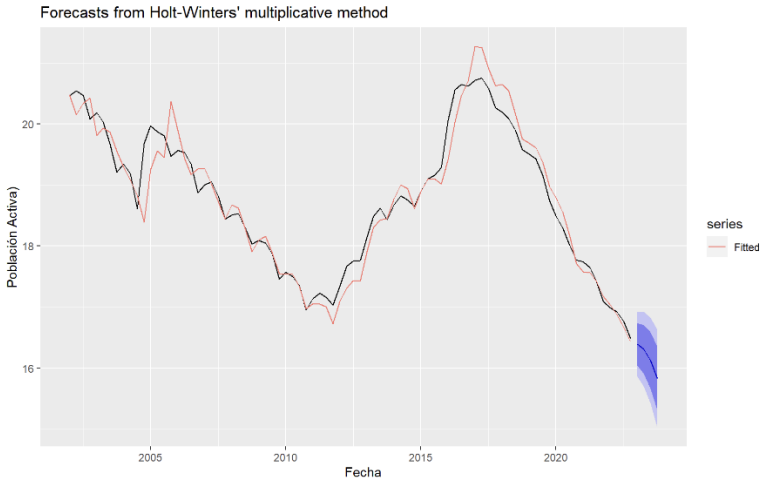
4. ALISADO

#Encontrar el modelo de suavizado exponencial más adecuado

Para nuestro caso particular, el mejor modelo de alisado es el de Holt Winters ya que nuestra serie tiene tanto tendencia como estacionalidad.

```
fit1 <- hw(Poblacion_TR, h=4, seasonal="multiplicative")
```

```
autoplot(fit1) +
    autolayer(fitted(fit1), series="Fitted") +
    ylab("Población activa") + xlab("Fecha")
```



En negro se presenta la serie original, en rojo se representa el ajuste que ha hecho el modelo y en azul las predicciones que realiza, con los intervalos de confianza en azul más oscuro y en más claro al 80% y al 95% respectivamente

Mostramos las predicciones en una tabla con el summary de nuestro modelo que acabamos de ajustar.

summary(fit1)

```
Forecast method: Holt-Winters' multiplicative method

Model Information:
Holt-Winters' multiplicative method

Call:
 hw(y = Poblacion_TR, h = 4, seasonal = "multiplicative")

  Smoothing parameters:
    alpha = 0.4863
    beta  = 0.0958
    gamma = 0.5137

  Initial states:
    l = 20.7421
    b = -0.2133
    s = 1.0099 1.0005 0.9921 0.9975

  sigma:  0.0164

     AIC     AICc      BIC
183.9960 186.4285 205.8734

Error measures:
                     ME      RMSE       MAE        MPE
Training set 0.004787631 0.3011481 0.2153814 0.03629509
                  MAPE      MASE       ACF1
Training set 1.120398 0.3523046 0.4854376

Forecasts:
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2023 Q1       16.38797 16.04268 16.73326 15.85989 16.91605
2023 Q2       16.30940 15.91074 16.70807 15.69970 16.91911
2023 Q3       16.12286 15.66396 16.58177 15.42103 16.82469
2023 Q4       15.82399 15.30084 16.34713 15.02391 16.62407
>
```

Se presentan los límites inferiores y superiores para ambos intervalores de confianza y para cada trimestre del año 2023 (predicción).

MODELO MATEMÁTICO

Modelo Multiplicativo

$$L_t = \alpha \frac{x_t}{S_{t-s}} + (1-\alpha)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}$$

$$S_t = \gamma \frac{x_t}{L_t} + (1-\gamma)S_{t-s}$$

modela la estacionalidad

$$\hat{x}_{t+1} = (L_t + b_t)S_{t+1-s}$$

$$L_t = \alpha \frac{x_t}{S_{t-1}} + (1-\alpha)(L_{t-1} + b_{t-1})$$

$$b_t = \beta(L_t - L_{t-1}) + (1-\beta)b_{t-1}$$

$$S_t = \gamma \frac{x_t}{L_t} + (1-\gamma)S_{t-s}$$

$$\hat{x}_{t+1} = (L_t + b_t)S_{t+1-s}$$

$$L_t = 0{,}4863 \frac{x_t}{S_{t-1}} + (1-0{,}4863)(L_{t-1} + b_{t-1})$$

$$b_t = 0{,}0958(L_t - L_{t-1}) + (1-0{,}0958)b_{t-1}$$

$$S_t = 0{,}5137 \frac{x_t}{L_t} + (1-0{,}5137)S_{t-s}$$

$$\hat{x}_{t+1} = (L_t + b_t)S_{t+1-s}$$

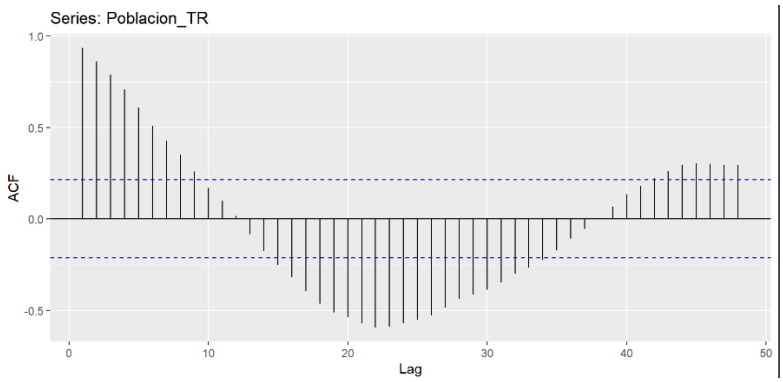
El periodo S para nuestro caso particular es 4. El primer valor que podemos calcular de L es L2 ya que no disponemos de los datos de la serie anteriores.

5.

#Representar las funciones de autocorrelación simple y parcial

Función de autocorrelación simple. Esta función nos permite estudiar y determinar tanto la estacionalidad como la estacionariedad de la serie.

ggAcf(Poblacion_TR, lag=48)



En este punto es importante tener un número de retardos (lag) suficientemente elevado para poder observar la estacionalidad de nuestra serie. Para este caso particular, un lag = 48 parece suficiente según lo observado en la imagen anterior.

Podemos observar también que la serie se muestra estacionaria, ya que sigue un patrón recurrente que en este caso se asemeja a una función sinusoidal. En cuanto a estacionariedad, la serie parece tener una media bastante constante en el tiempo cancelándose los instantes de tiempo en la que presenta valores negativos con los positivos. De todas maneras, no podemos afirmar que la serie sea estacionaria en media hasta observar los valores necesarios que serán obtenidos más adelante en el desarrollo.

Para observar numéricamente los valores mostrados utilizamos la función:

acf(Poblacion_TR, lag.max =20, plot =F)

```
Autocorrelations of series 'Poblacion_TR', by lag

  0.00   0.25   0.50   0.75   1.00   1.25   1.50   1.75   2.00   2.25   2.50   2.75   3.00   3.25
 1.000  0.935  0.860  0.788  0.709  0.607  0.509  0.428  0.350  0.259  0.169  0.098  0.016 -0.083
  3.50   3.75   4.00   4.25   4.50   4.75   5.00
-0.176 -0.253 -0.317 -0.395 -0.463 -0.509 -0.538
```

Gracias a esta función obtenemos los valores de los retardos (lag). Por ejemplo, el valor del retardo 1 vale 0.935, el 2 vale 0.860 etc. Estos valores indican la correlación que existe entre la serie original y la serie retardada un instante i. Es decir, mide la relación lineal entre la serie en el instante t y en t-i. A modo de ejemplo, en nuestro caso particular, el valor 0.935representa la relación entre el valor de la serie en el instante t y en el t-1, y el valor 0.860 es la correlación entre los valores de la serie en el instante t y en el t-2, es decir, cuando la serie se encuentra retardada 2 instantes.

Realizando el procedimiento análogo para la función de autocorrelación parcial con la función,

pacf(Poblacion_TR, lag.max =20, plot =F)

obtenemos:

```
Partial autocorrelations of series 'Poblacion_TR', by lag

  0.25   0.50   0.75   1.00   1.25   1.50   1.75   2.00   2.25   2.50   2.75   3.00   3.25   3.50
 0.935 -0.115 -0.005 -0.109 -0.215 -0.016  0.058 -0.016 -0.156 -0.064  0.036 -0.177 -0.163 -0.060
  3.75   4.00   4.25   4.50   4.75   5.00
-0.029  0.038 -0.165 -0.073 -0.025  0.045
```

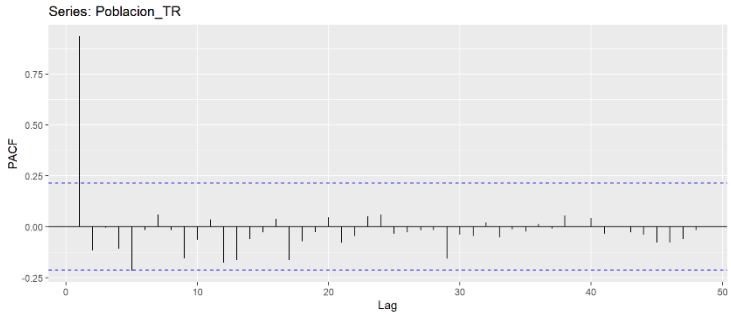
Observamos que los valores difieren considerablemente, no es alarmante ya que esta función de autocorrelación parcial mide la relación entre las variables eliminada la influencia que existe entre los retardos intermedios. Teniendo en cuenta esto, es normal que el primer valor

de retardo 1 coincida con el de la función de autocorrelación simple porque no hay retardos intermedios al tratarse del primer valor, en cambio, el valor 2 si que es diferente porque se ha eliminado la influencia del retardo t-1 existente entre el valor 1 y el 2.

Función de autocorrelación parcial.

No nos arroja información sobre el comportamiento de la serie como hace la función de autocorrelación simple, en cambio, esta función se utiliza para determinar los órdenes del modelo ARIMA.

ggPacf(Poblacion_TR,lag=48)



#Ajustar el modelo ARIMA adecuado

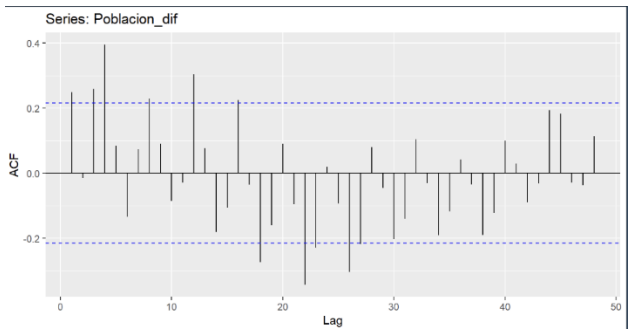
#Representar las ACF y PACF para la serie con las diferenciaciones para ver los parámetros del modelo ARIMA:

Como no sabemos con certeza si la serie es estacionaria en media, vamos en primer lugar a suponer que no lo es y observar que ocurre. Para este caso hacemos la diferenciación(diferenciación de una serie es un método que consiste en no hacer ninguna hipótesis sobre la forma de la tendencia a corto plazo y suponer simplemente que evoluciona lentamente en el tiempo) de la serie con la función:

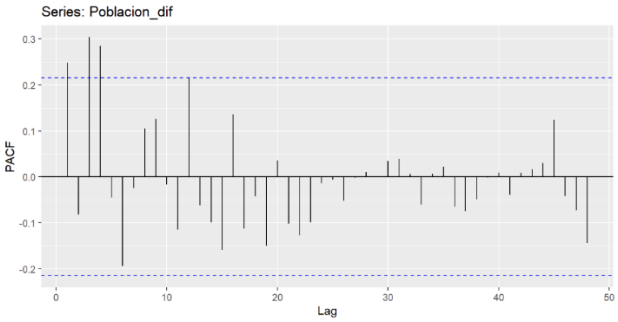
Poblacion_dif<-diff(Poblacion_TR)

Ahora calculamos la función de autocorrelación y de autocorrelación parcial respectivamente.

ggAcf(Poblacion_dif, lag=48)



ggPacf(Poblacion_dif, lag=48)



A la vista de estos resultados debemos determinar los órdenes p (orden de la parte autorregresiva) y q (orden de la parte de media móvil). Para observar los valores P nos centramos en la función de autocorrelación parcial y observamos los primeros retardos que son significativos, es decir, que se salen de las bandas. En vista de que los primeros valores significativos se salen tanto de las bandas es un síntoma de que nos hemos equivocado de interpretación y de que la serie si que es estacionaria en media y por lo tanto construiremos otro modelo sin hacer la diferenciación.

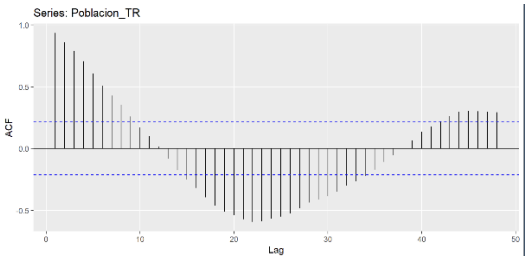
Vamos a ajustar el primer modelo ARIMA suponiendo que nuestra serie no sea estacionaria en media, es decir, nuestro primer modelo se trata de un I1.

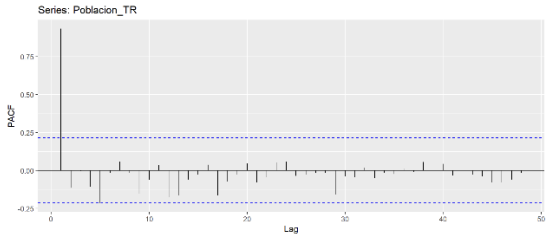
fitARIMA<-arima(Poblacion_TR,order=c(0,1,0))

summary(fitARIMA)

```
Call:
arima(x = Poblacion_TR, order = c(0, 1, 0))


sigma^2 estimated as 0.07269:  log likelihood = -8.98,  aic = 19.96

Training set error measures:
                     ME      RMSE       MAE        MPE     MAPE     MASE
Training set -0.04714925 0.2680112 0.2049698 -0.2664387 1.093746 0.989271
                    ACF1
Training set 0.2501581
```

Otro posible modelo es considerar que es estacionaria en media, por tanto no hacer la diferenciación de la serie y entonces determinamos los ordenes de los parámetros correspondientes de la parte autorregresiva y la parte de media móvil calculando la función de autocorrelación y autocorrelación parcial de la serie original y partir de ahí determinamos los parámetros.





Se observa en PACF el primer retardo significativo y el resto 0 lo cual es síntoma de un patrón AR (autorregresivo) . Se trata en particular de un patrón AR1 porque en la función de autocorrelación decaen los primeros retardos. Para comprobar esta afirmación de forma numérica utilizamos la función:

fitARIMA_1<-arima(Poblacion_TR,order=c(1,0,0))

summary(fitARIMA_1)

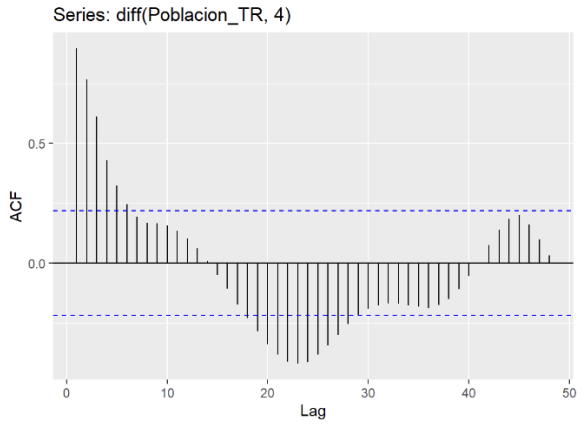
```
Call:
arima(x = Poblacion_TR, order = c(1, 0, 0))

Coefficients:
      ar1  intercept
        1    18.6679
s.e.  NaN   568.4431

sigma^2 estimated as 0.07183:  log likelihood = -9.09,  aic = 24.17

Training set error measures:
                      ME      RMSE       MAE        MPE     MAPE     MASE      ACF1
Training set -0.04738752 0.2680019 0.2047315 -0.2676031 1.092581 0.988121 0.2498216
```

Para estimar que modelo es mejor nos fijamos en la aic, aic(modelo1) = 19.96 y aic(modelo2) = 24.17 por tanto aic(modelo2) > aic(modelo1) lo que implica que el mejor modelo según el criterio aic es el segundo, en el que no hacemos la diferenciación.

Modeleo3. Como no estamos seguros de que la serie sea estacional en media, un posible modelo sería asumir que no lo sea. Partiendo de esta premisa, vamos a convertir la serie en estacionaria y eliminar la posible es estacionalidad.

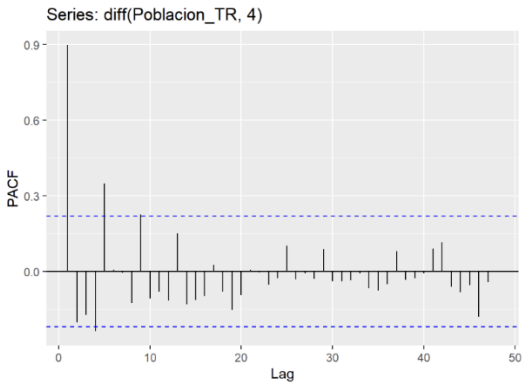
Asumiendo que no sea estacionaria en media, la forma para elegir la diferenciación que tendremos que hacerle a la serie para convertirla en estacionaria en media, es la diferenciación regular. Y para desestacionalizarla, la diferenciación la hacemos de orden 4 que es el periodo de nuestra estacionalidad.

ggAcf(diff(Poblacion_TR,4), lag=48)



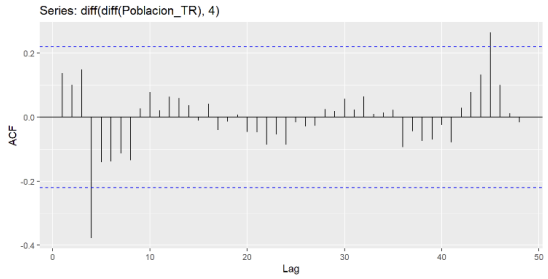
Observamos un decaimiento lento en los retardos, síntoma de que la serie no es estacionaria en media.

ggPacf(diff(Poblacion_TR,4), lag=48)

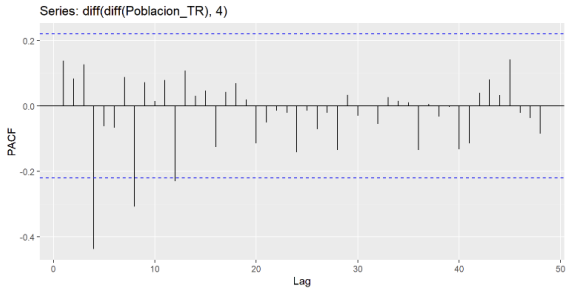


Hacemos la ACF con las dos diferenciaciones, la regular y la estacional

ggAcf(diff(diff(Poblacion_TR),4), lag=48)



ggPacf(diff(diff(Poblacion_TR),4), lag=48)



Como es de esperar no se observa estacionalidad porque no vemos decaimientos lentos de los retardos en los periodos, si los hubiese, tendríamos que hacer una diferenciación mas.

Teniendo en cuenta estas consideraciones, podemos determinar que modelo ajustamos y calcular su idoneidad. Primero observamos los primeros retardos de la parte regular de ambas funciones. Ambas presentan retrasos significativos. En la función de autocorrelación tenemos un retraso significativo en la posición 4 y el resto irrelevantes y para la parcial lo mismo. Por lo tanto, para la parte regular el modelo es

NO tenemos ningún parámetro ni en la parte regresiva ni en la de media móvil.

fitARIMA<-arima(Poblacion_TR,order=c(0,1,0), seasonal=c(0,1,1))

En la parte estacional nos tenemos que centrar en los retardos múltiplos de 4. Observando el grafico de PACF, observamos retardos en 4 y en la de ACF lo mismo por tanto no tenemos patrones claros.

fitARIMA<-arima(Poblacion_TR,order=c(0,1,0), seasonal=c(0,1,1))

```
> summary(fitARIMA)

Call:
arima(x = Poblacion_TR, order = c(0, 1, 0), seasonal = c(0, 1, 1))

Coefficients:
          sma1
      -0.7503
s.e.   0.1538

sigma^2 estimated as 0.0606:  log likelihood = -3.01,  aic = 10.02

Training set error measures:
                      ME      RMSE       MAE          MPE      MAPE      MASE      ACF1
Training set -0.00813384 0.2387759 0.1569512 -0.03959456 0.826662 0.7575129 0.3106645
```

fitARIMA_1<-arima(Poblacion_TR,order=c(1,1,1), seasonal=c(0,1,1))

```
> summary(fitARIMA_1)

Call:
arima(x = Poblacion_TR, order = c(1, 1, 1), seasonal = c(0, 1, 1))

Coefficients:
         ar1      ma1     sma1
      0.8721  -0.6187  -1.0000
s.e.  0.1081   0.1584   0.1342

sigma^2 estimated as 0.04474:  log likelihood = 5.04,  aic = -2.07

Training set error measures:
                       ME      RMSE       MAE           MPE      MAPE      MASE        ACF1
Training set -0.0005565021 0.2051835 0.1233547 0.0005199185 0.6519561 0.5953622 0.005807307
```

fitARIMA_2<-arima(Poblacion_TR,order=c(0,1,0), seasonal=c(0,1,3))

```
> summary(fitARIMA_2)

Call:
arima(x = Poblacion_TR, order = c(0, 1, 0), seasonal = c(0, 1, 3))

Coefficients:
          sma1     sma2     sma3
      -0.7526  -0.2338  -0.0137
s.e.   0.1589   0.1499   0.1116

sigma^2 estimated as 0.05247:  log likelihood = -0.97,  aic = 9.94

Training set error measures:
                      ME      RMSE       MAE         MPE      MAPE      MASE      ACF1
Training set 0.003634105 0.2221883 0.1423873 0.01724641 0.7518101 0.6872214 0.280064
```

Basándonos en el criterio de aic parece que el mejor modelo es el segundo por que su aic es mayor, de todas maneras este no es un método de selección demasiado fiable, vamos a analizar otros factores relevantes para seleccionar el mejor modelo.

Comprobar la idoneidad del modelo

coeftest(fitARIMA)

coeftest(fitARIMA_1)

coeftest(fitARIMA_2)

```
> coeftest(fitARIMA)

z test of coefficients:

     Estimate Std. Error z value  Pr(>|z|)
sma1 -0.75032    0.15376 -4.8796 1.063e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(fitARIMA_1)

z test of coefficients:

     Estimate Std. Error z value  Pr(>|z|)
ar1   0.87212    0.10806  8.0706 6.995e-16 ***
ma1  -0.61868    0.15842 -3.9052 9.413e-05 ***
sma1 -0.99996    0.13418 -7.4523 9.173e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> coeftest(fitARIMA_2)

z test of coefficients:

      Estimate Std. Error z value  Pr(>|z|)
sma1 -0.752576   0.158926 -4.7354 2.186e-06 ***
sma2 -0.233767   0.149922 -1.5593    0.1189
sma3 -0.013656   0.111630 -0.1223    0.9026
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A primera vista parece que el mejor modelo es el ARIMA 1 porque los tres factores (AR1,MA1 y media móvil) son significativos.

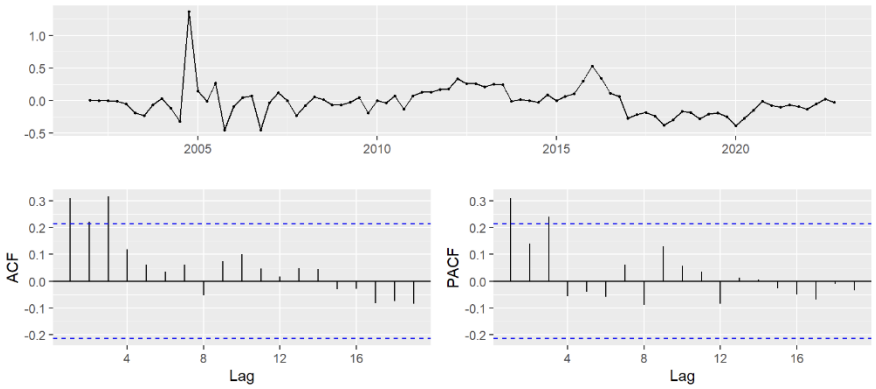
Comprobar hipótesis dadas a los residuos

resi<-residuals(fitARIMA)

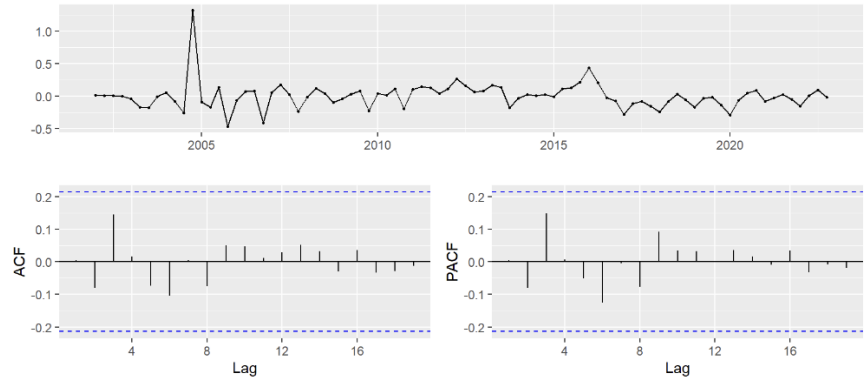
resi_1<-residuals(fitARIMA_1)

resi_2<-residuals(fitARIMA_2)

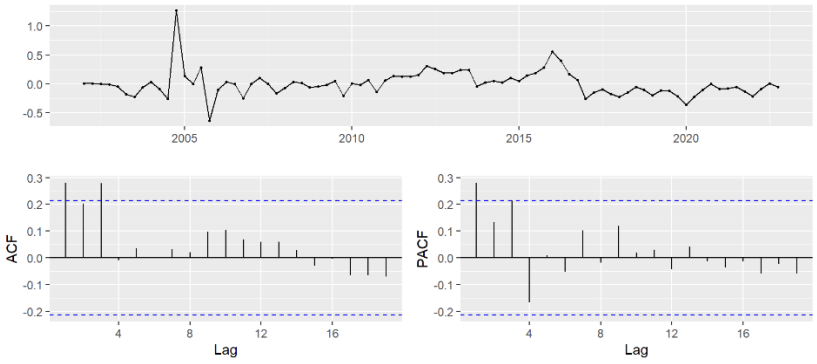
ggtsdisplay(resi)



ggtsdisplay(resi_1)

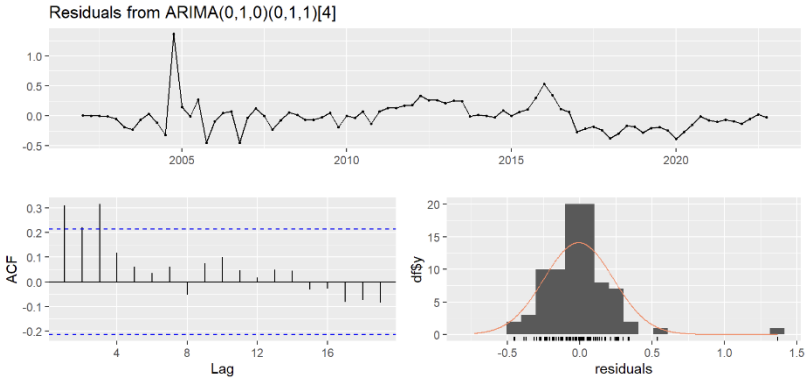


ggtsdisplay(resi_2)



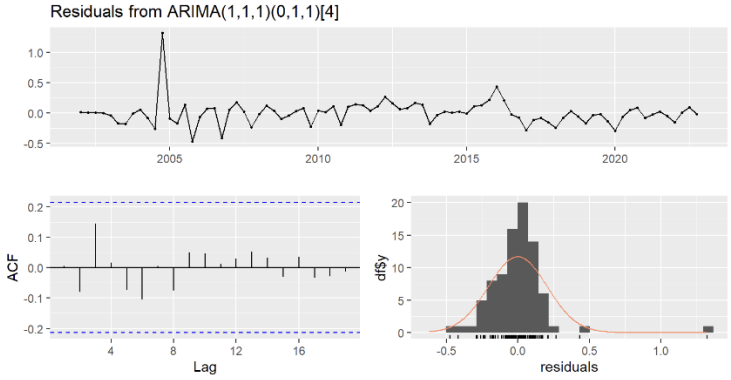
Analizando los gráficos de los residuos, el único modelo que no presenta retardos significativos es el segundo, que de nuevo parece ser el mejor.

checkresiduals(fitARIMA)



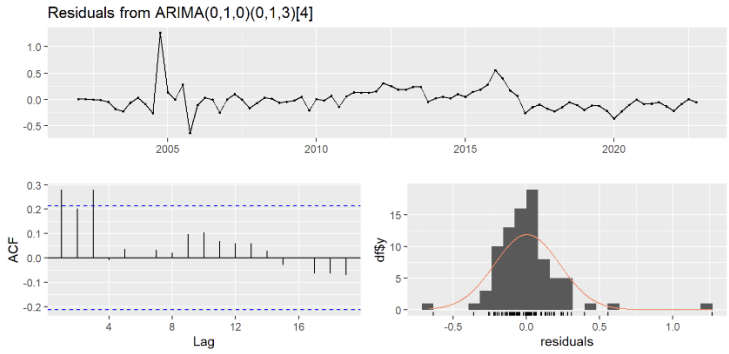

```
> checkresiduals(fitARIMA)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)(0,1,1)[4]
Q* = 23.94, df = 7, p-value = 0.001167

Model df: 1.   Total lags used: 8
```

checkresiduals(fitARIMA_1)



```
> checkresiduals(fitARIMA_1)

        Ljung-Box test

data:  Residuals from ARIMA(1,1,1)(0,1,1)[4]
Q* = 4.5153, df = 5, p-value = 0.4778

Model df: 3.   Total lags used: 8
```

checkresiduals(fitARIMA_2)




```
> checkresiduals(fitARIMA_2)

        Ljung-Box test

data:  Residuals from ARIMA(0,1,0)(0,1,3)[4]
Q* = 17.658, df = 5, p-value = 0.003407

Model df: 3.   Total lags used: 8
```

Analizando el p-valor, se observa la correlación entre los residuos y una vez más el modelo que presenta mayor correlación es el segundo, con un p-valor de 0.4778.

Por ultimo podemos estudiar las medidas de error en el ajuste, es decir, con los datos que hemos utilizado para estimar. Esto se consigue con la función accuracy:

accuracy(fitARIMA)

```
> accuracy(fitARIMA)
                     ME      RMSE       MAE          MPE     MAPE      MASE      ACF1
Training set -0.00813384 0.2387759 0.1569512 -0.03959456 0.826662 0.7575129 0.3106645
>
```

accuracy(fitARIMA_1)

```
> accuracy(fitARIMA_1)
                      ME      RMSE       MAE         MPE      MAPE      MASE        ACF1
Training set -0.0005565021 0.2051835 0.1233547 0.0005199185 0.6519561 0.5953622 0.005807307
```

accuracy(fitARIMA_2)

```
> accuracy(fitARIMA_2)
                     ME      RMSE       MAE         MPE      MAPE      MASE      ACF1
Training set 0.003634105 0.2221883 0.1423873 0.01724641 0.7518101 0.6872214 0.280064
```

Basándonos en el análisis del RMSE el mejor modelo sería el segundo por que es menor, 0.2051.

En vista de estos resultados, he decidido que el mejor modelo es el segundo.

Además de comprobar como de bueno ha sido el ajuste, tenemos que comprobar como de bien predice nuestro modelo ganador (el 2)

Calculamos las predicciones del modelo ganador (ARIMAA1)

con la siguiente función

forecast(fitARIMA_1,h=4)

```
> forecast(fitARIMA_1,h=4)
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2023 Q1       16.55637 16.27880 16.83395 16.13186 16.98089
2023 Q2       16.53591 16.09136 16.98047 15.85602 17.21580
2023 Q3       16.34214 15.73865 16.94563 15.41918 17.26510
2023 Q4       16.09853 15.33907 16.85798 14.93704 17.26001
```

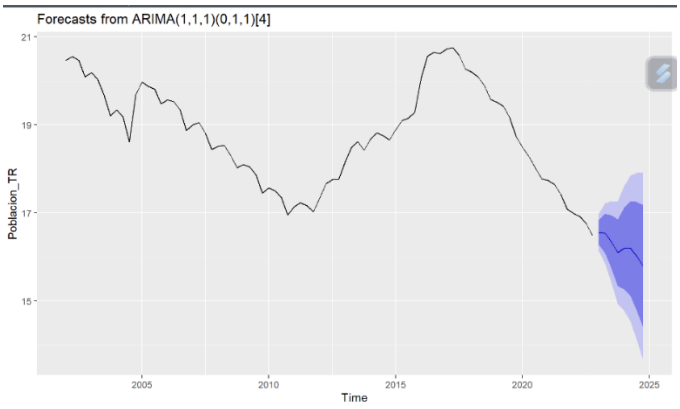
summary(forecast(fitARIMA_1,h=4))

```
Forecast method: ARIMA(1,1,1)(0,1,1)[4]

Model Information:

Call:
arima(x = Poblacion_TR, order = c(1, 1, 1), seasonal = c(0, 1, 1))

Coefficients:
          ar1      ma1     sma1
      0.8721  -0.6187  -1.0000
s.e.  0.1081   0.1584   0.1342

sigma^2 estimated as 0.04474:  log likelihood = 5.04,  aic = -2.07

Error measures:
                      ME      RMSE       MAE          MPE      MAPE      MASE
Training set -0.0005565021 0.2051835 0.1233547 0.0005199185 0.6519561 0.2017743
                    ACF1
Training set 0.005807307

Forecasts:
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2023 Q1       16.55637 16.27880 16.83395 16.13186 16.98089
2023 Q2       16.53591 16.09136 16.98047 15.85602 17.21580
2023 Q3       16.34214 15.73865 16.94563 15.41918 17.26510
2023 Q4       16.09853 15.33907 16.85798 14.93704 17.26001
```

Con el summary de la función anterior se obtienen los valores de los intervalos de confianza para los cuatro trimestres del año 2023 que es nuestra predicción.

Representamos las previsiones del modelo ganador

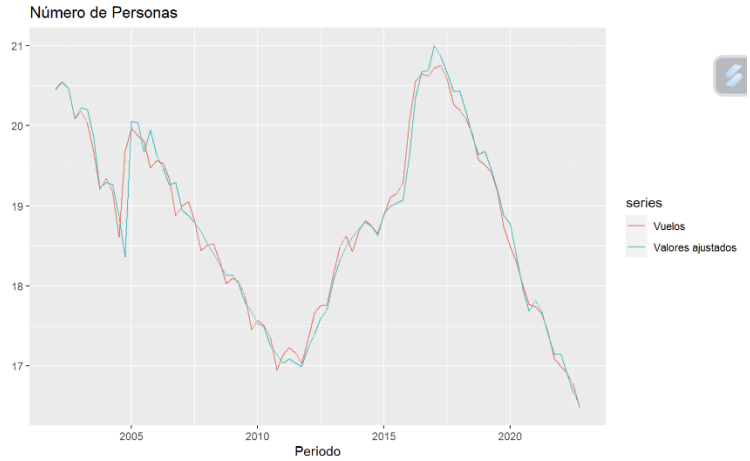
summary(forecast(fitARIMA_1,h=4))



Vamos a repredsentar los valores observados de nuestra serie original, superpuestos con las calores ajustados para ver como de bueno ha sido el ajuste observación a observación.

cbind("Vuelos" = Poblacion_TR, "Valores ajustados" =fitted(fitARIMA_1)) %>%

autoplot() + xlab("Periodo") + ylab("") + ggtitle("Número de Personas")



Y vemos que el modelo se ajusta bastante bien a los datos obervados.

8. Comparar los modelos de Holt-Winters y ARIMA1

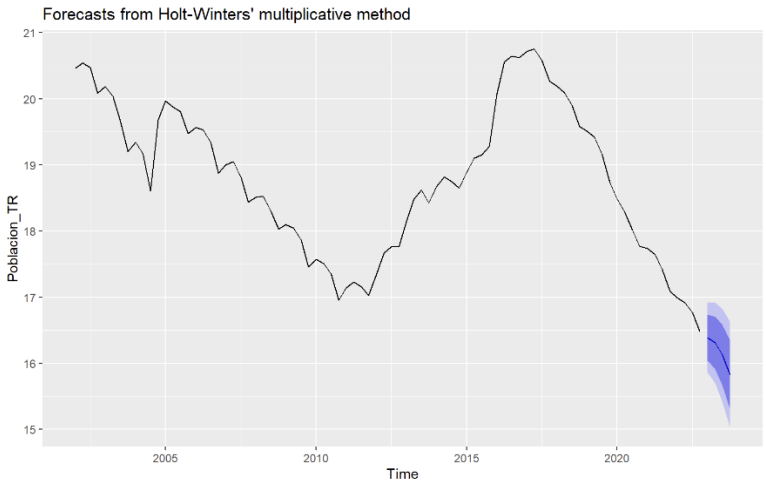
La forma de compararlos será medir la diferencia entre el valor real y estimado de los últimos valores de la serie, justo los que no han sido utilizados para estimar el modelo.

#Comprobar el ajuste del modelo ARIMA y del modelo de suavizado exponencial

Modelo de suavizado exponencial. Holt-Winter porque tiene estacionalidad

Recordamos de nuevo la grafica de la serie con la predicción obtenida por HW.

fit1 <- hw(Poblacion_TR,h=4, seasonal="multiplicative")



Y observamos las medidas de ajuste que realiza cada modelo.

accuracy(fitARIMA_1)

```
                      ME      RMSE       MAE           MPE      MAPE      MASE        ACF1
Training set -0.0005565021 0.2051835 0.1233547 0.0005199185 0.6519561 0.5953622 0.005807307
>
```

accuracy(fit1)

```
                      ME      RMSE       MAE        MPE     MAPE      MASE      ACF1
Training set 0.004787631 0.3011481 0.2153814 0.03629509 1.120398 0.3523046 0.4854376
```

Si nos centramos en la medida de error RMSE de los dos modelos, observamos que el más elevado es el de HW (0.30) y por lo tanto el mejor modelo predictivo es el ARIMA1, con un RMSE menor (0.20)

Por ultimo nos interesa saber como de buenas han sido las predicciones que nos ha hecho el modelo, si el modelo de alisado exponencial llevado a cabo en el modelo de HW o los métodos ARIMA. Para ello vamos a comparar los valores reales de la serie, que los tenemos guardados, conlos valores que ha predicho. Lo que vamos a hacer con estos valores reales es compararlos con los predichos en ese mismo periodo y así ver la divergencia.

Estos datos que nos hemos guardado con los últimos valores de la serie son:

Poblacion_RealPredicho<-window(Poblacion,start=c(2002,1), end=c(2022,4))

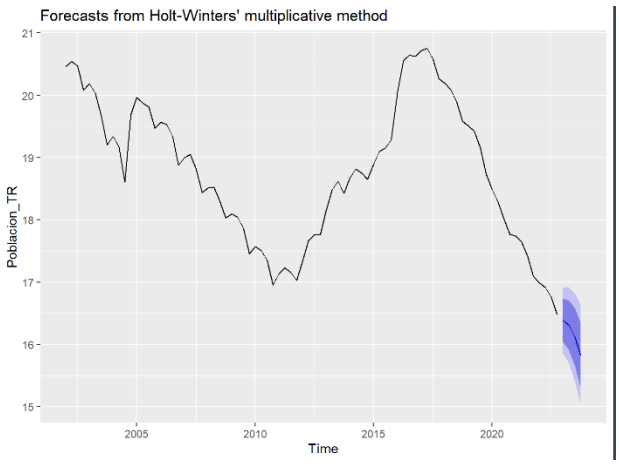
Este periodo ya se había estimado con anterioridad.

Nos guardamos las predicciones obtenidas con el modelo ARIMA en la siguientes variable.

pred<-forecast(fitARIMA_1,h=4)

y de la variable pred nos quedamos solamente con los valores de la predicción que se guardan en mean.

valores<-pred$mean



Representamos los valores eales frente a los predichos pero solo de los últimos 4 valores que hemos ajustado

Poblacion_RealPredicho<-window(Poblacion,start=c(2022,1), end=c(2022,4))

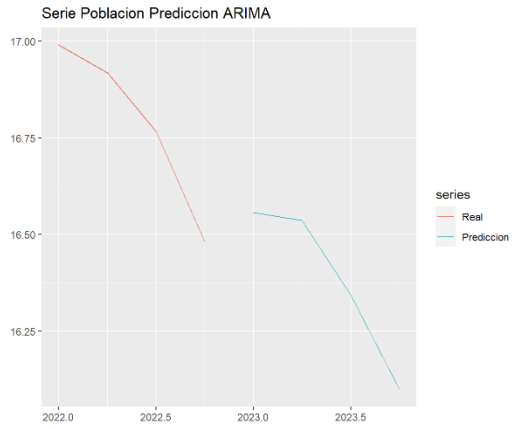
Guardamos las predicciones obtenidas con el modelo ARIMA

pred<-forecast(fitARIMA_1,h=4)

valores<-pred$mean

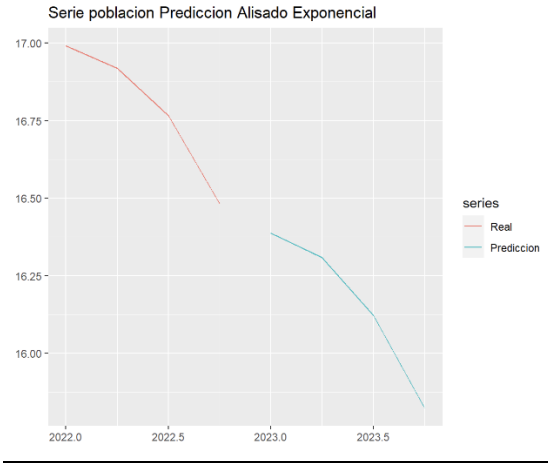
Representamos los valores reales frente a los estimados

```
cbind("Real" = Poblacion_RealPredicho, "Prediccion" =valores) %>%

  autoplot() + xlab("") + ylab("") +   ggtitle("Serie Poblacion Prediccion ARIMA")
```



Hacemos lo mismo con las predicciones del modelo HW

```
valores_Alisado<-fit1$mean

#Representamos los valores reales frente a los estimados

cbind("Real" = Poblacion_RealPredicho, "Prediccion" =valores_Alisado) %>%

  autoplot() + xlab("") + ylab("") +   ggtitle("Serie poblacion Prediccion Alisado Exponencial")
```



Viendo estos resultados me he dado cuenta de que acorté mal la serie original, y por eso no me reservé los últimos valores correctamente y por ello salen las gráficas desplazadas y no solapadas, de todas maneras se aprecia que la tendencia es clara y si desplazamos una de las series para solaparlas se ajustan muy bien. En conclusión los predictivos ARIMA1 y HW se ajustan de forma muy precisa a la serie original. Diría que el mejor es el de HW ya que la tendencia predictiva se ajusta mejor a la serie original y como hemos visto, pese a que el valor de error sea mayor que el de ARIMA1, la diferencia era del orden de 0.1.