Ejercicio 7

1. Utiliza el conjunto datos PBI.txt de la tarea 1 y crear un nuevo conjunto de datos llamado PBI_1 que conste de los mismos datos que el primer .txt pero sin los últimos 6 valores.

Primero, cargamos nuestros datos:

```
datos <- read_xls("PIB.xls")
head(datos)
```

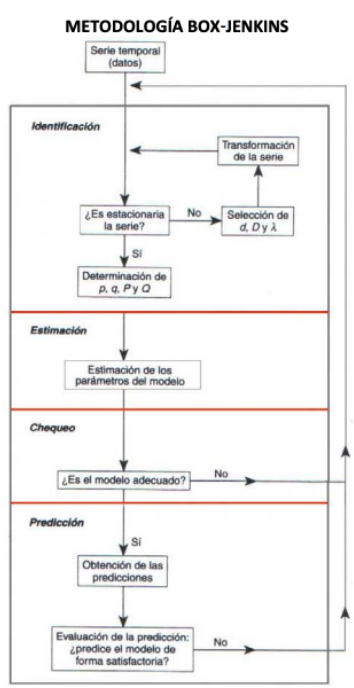
```
## # A tibble: 6 x 2
##   ‘Instituto Nacional de Estadística y Geografía (INEGI)‘ ...2
##   <chr>                                                   <chr>
## 1 Banco de Información Económica (BIE)                    <NA>
## 2 Fecha de consulta: 06/09/2023 12:33:45                  <NA>
## 3 <NA>                                                    <NA>
## 4 Periodos                                                Cuentas nacionales > ~
## 5 1980/01                                                 10401367.607000001
## 6 1980/02                                                 10342350
```

Ahora, declaramos la base de datos en serie de tiempo, donde eliminaremos los últimos 6 valores.

```
datos1<-datos[-c(1,2,3,4),]
datos1$...2<-as.numeric(datos1$...2)
datos2<-datos1$...2
PIB_1<-datos[-c(1,2,3,4,173,174,175,176,177,178),]
PIB_1$...2<-as.numeric(PIB_1$...2)
# Convertimos los datos en una serie de tiempo
pib<-ts(PIB_1$...2, frequency = 4, start = c(1980,1))
```

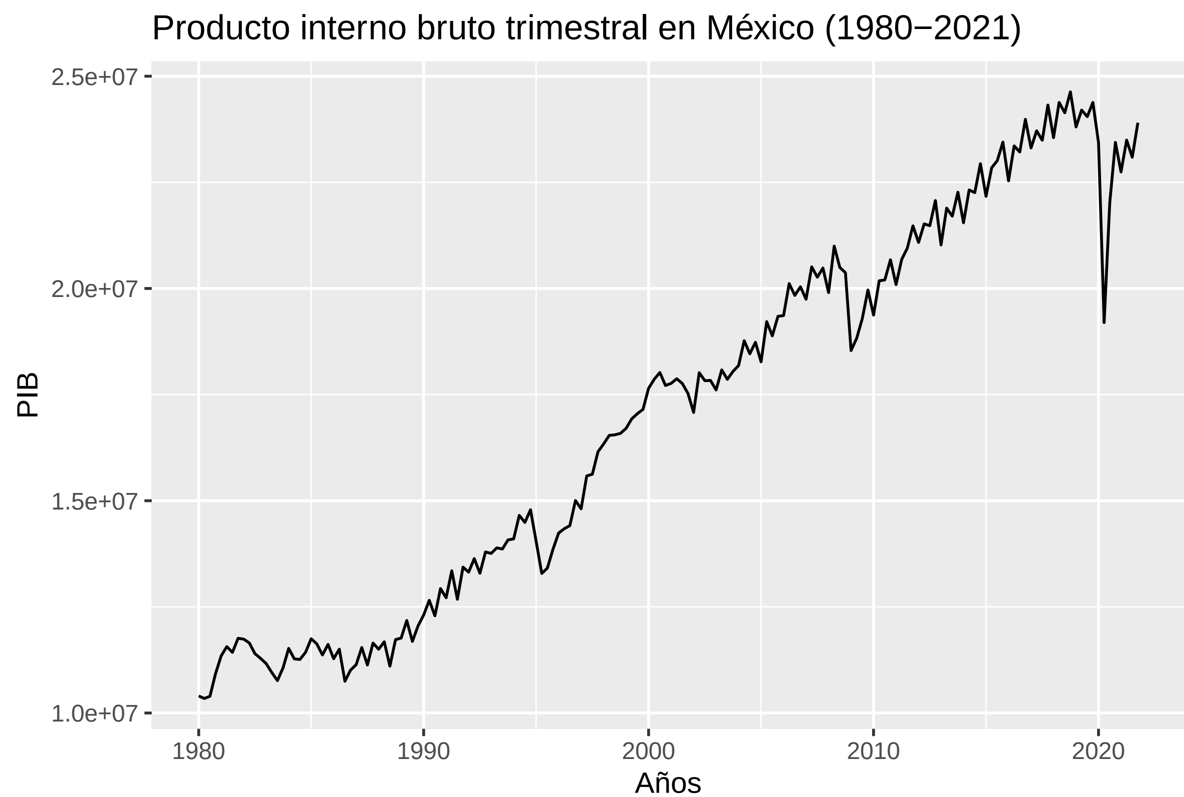
a) Encuentra un modelo ARIMA usando Metodología de Box-Jenkins. Tu análisis debería incluir una explicación lógica de los pasos que diste para quedarte con el modelo elegido.

Veamos que la metodología de Box-Jenkins nos indica que se deben de realizar los siguientes pasos:



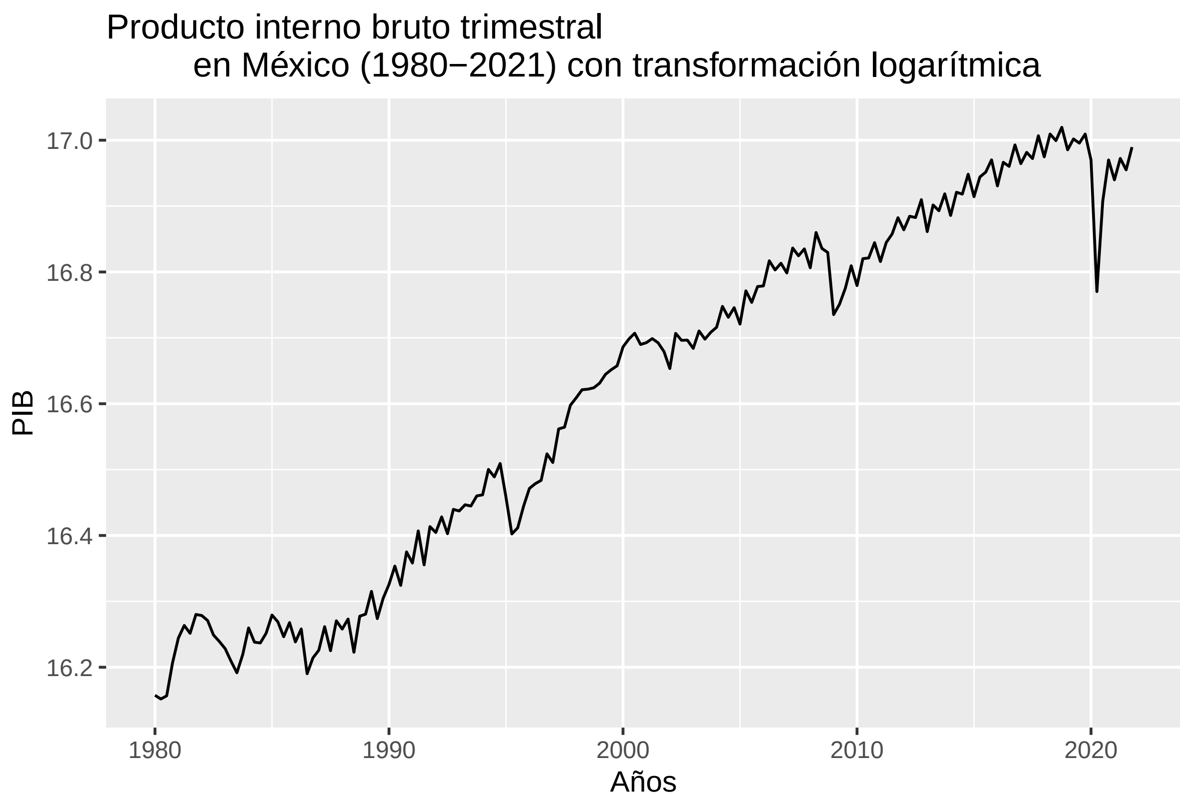
Primero, graficamos la serie de tiempo para visualizar su tendencia o ciclicidad:

```
autoplot(pib,ylab = "PIB", xlab = "Años", main = "Producto interno bruto trimestral en México (1980-2021)")
```



Obtenemos el logaritmo natural de los datos para hacer un escalamineto de los datos, ya que vemos que nuestros datos son en millones de pesos, es decir, son bastante grandes, por lo que, se hace un escalamiento de los datos para tener un mejor procesamiento:

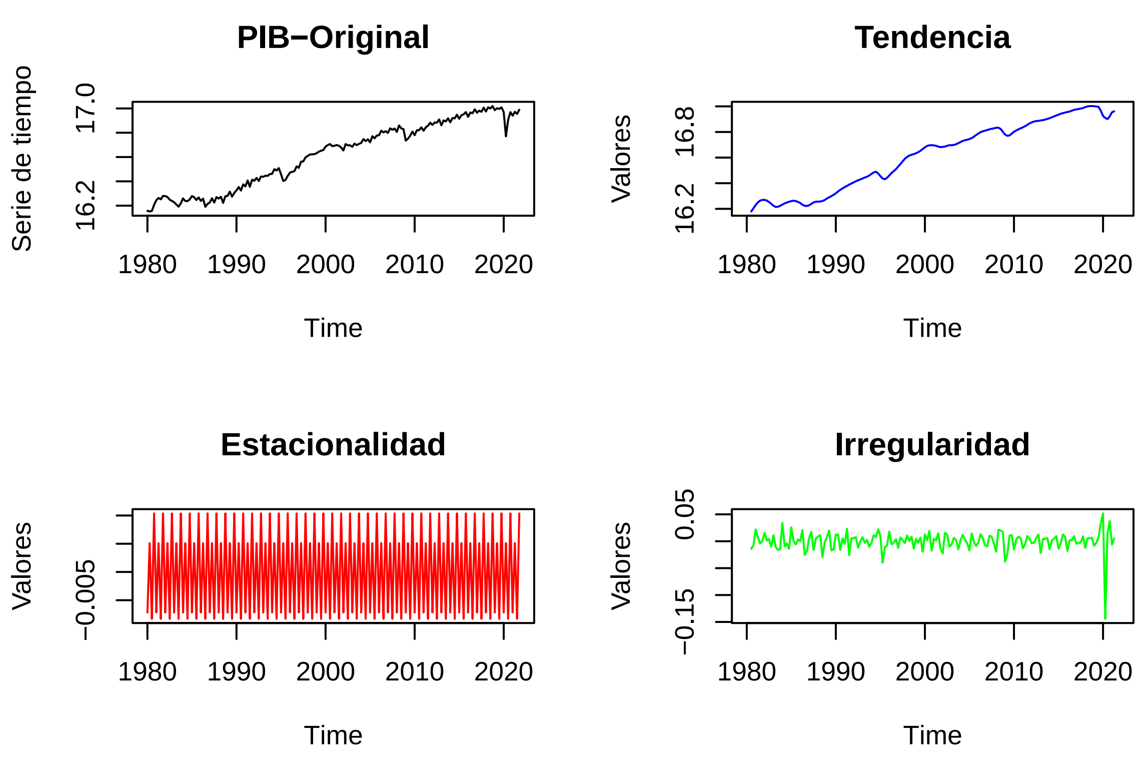
```
log_PIB_1<-log(PIB_1$...2)
# Convertimos los datos en una serie de tiempo
log_pib<-ts(log_PIB_1, frequency = 4, start = c(1980,1))
autoplot(log_pib,ylab = "PIB", xlab = "Años", main = "Producto interno bruto trimestral
         en México (1980-2021) con transformación logarítmica")
```



Procedemos a analizar sus patrones por medio de una descomposición:

```
#Utilizamos la función decompose (del paquete cargado previamente "STATS")
pib_decomp <- decompose(log_pib)

# Graficar los componentes
par(mfrow = c(2, 2)) #Se utiliza para dividir la ventana gráfica en una matriz de 2 filas y 2 columnas
plot(pib_decomp$x, main = "PIB-Original", col = "black", ylab = "Serie de tiempo")
plot(pib_decomp$trend, main = "Tendencia", col = "blue", ylab = "Valores")
plot(pib_decomp$seasonal, main = "Estacionalidad", col = "red", ylab = "Valores")
plot(pib_decomp$random, main = "Irregularidad", col = "green", ylab = "Valores")
```



Observamos que presenta tendencia creciente y un componente estacional.

```r
ggseasonplot(log_pib,year.labels = TRUE, year.labels.left = TRUE,ylab="PIB",xlab="",main="PIB trimestral en México
```

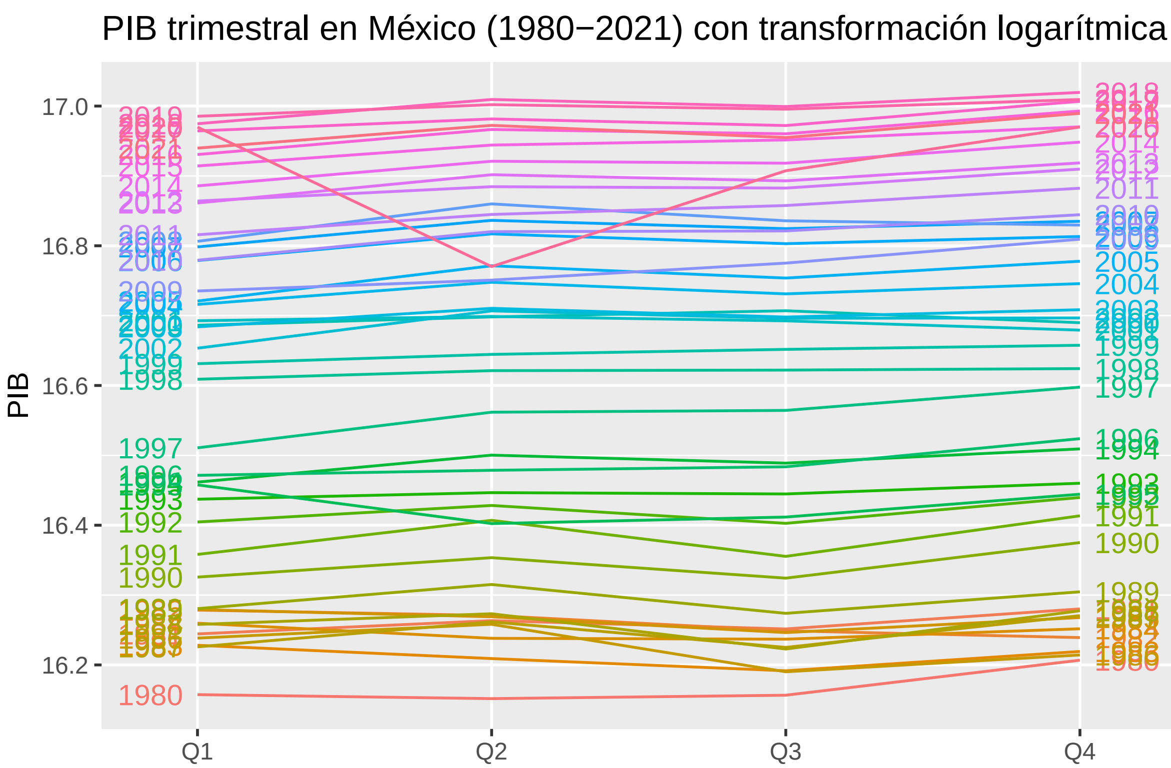


Figura 1: Visualización del componente estacional

Como podemos observar en la figura 1, podemos ver en casi todos los años, hay un crecimineto del PIB en Q_4.

Etapa 1: Identificación

Veremos si la serie es estacionaria por medio de la prueba del **Test de Dickey Fuller**.

Este test se basa en una regresión lineal que incluye la propia serie de tiempo y sus lags.

Las hipótesis respectivas son:

Contraste de hipótesis:

H0: Serie No estacionaria: Hay raiz unitaria **H1:** Serie Estacionaria: No hay raiz unitaria

```r
adf.test(log_pib)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log_pib
## Dickey-Fuller = -2.3265982, Lag order = 5, p-value = 0.4399639
## alternative hypothesis: stationary
```

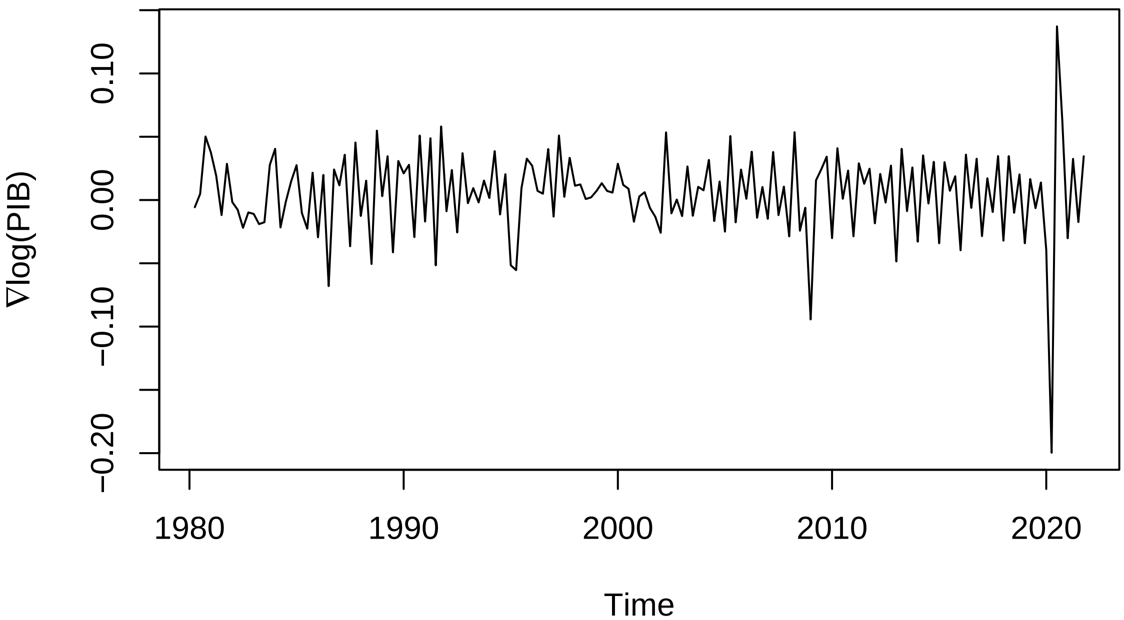
Tras realizar la prueba aumentada de Dickey-Fuller (ADF), obtenemos un *p-value = 0.4399*.

Como el *p-value* > 0.05, no rechazamos H0. Podemos concluir que nuestra serie de tiempo es NO Estacionaria, por lo que procedemos a diferenciar.

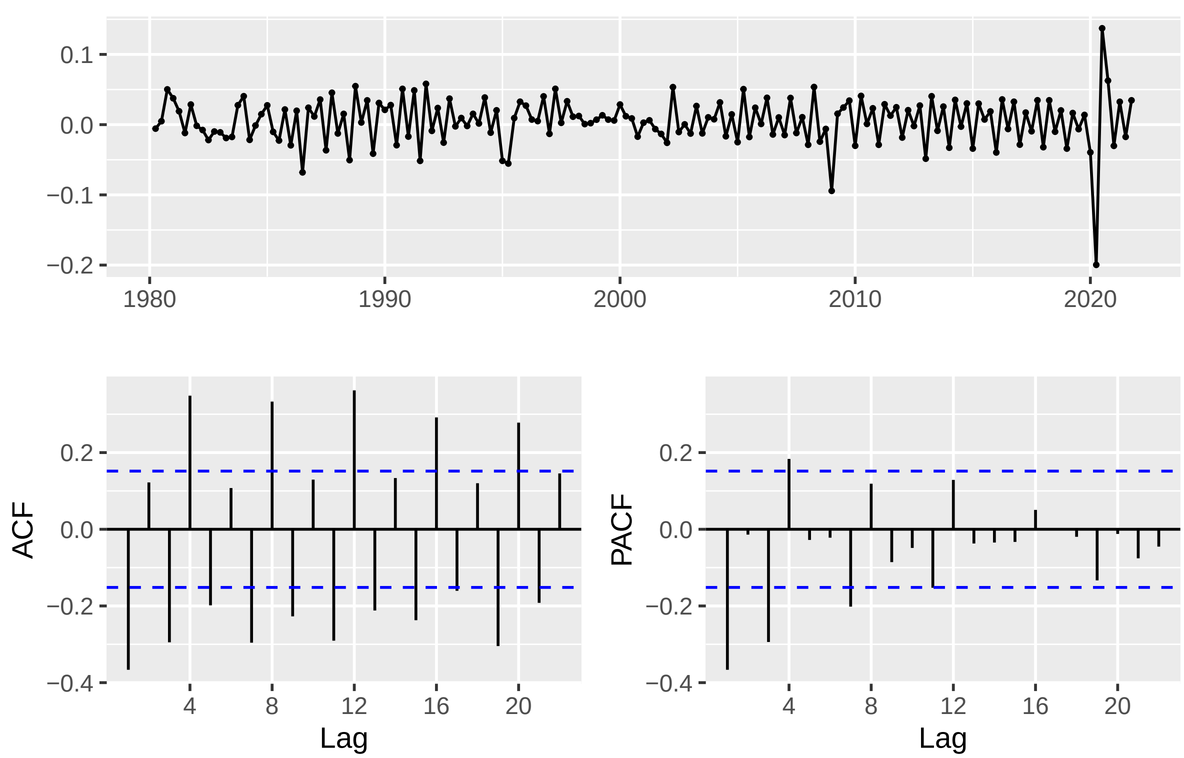
Aplicamos una diferencia para quitarle la tendencia lineal que parece tener nuestra serie.

```r
log_pib_d1 <- diff(log_pib, differences = 1)

plot(log_pib_d1, ylab = expression(paste(nabla, "log(PIB)")))
```

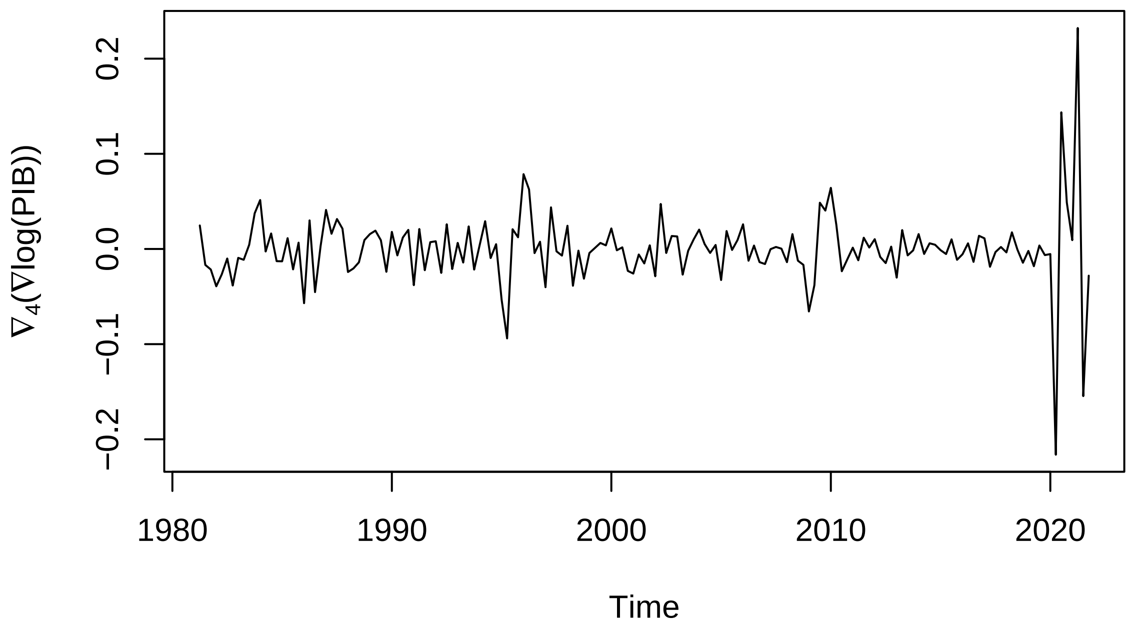



```r
ggtsdisplay(log_pib_d1)
```

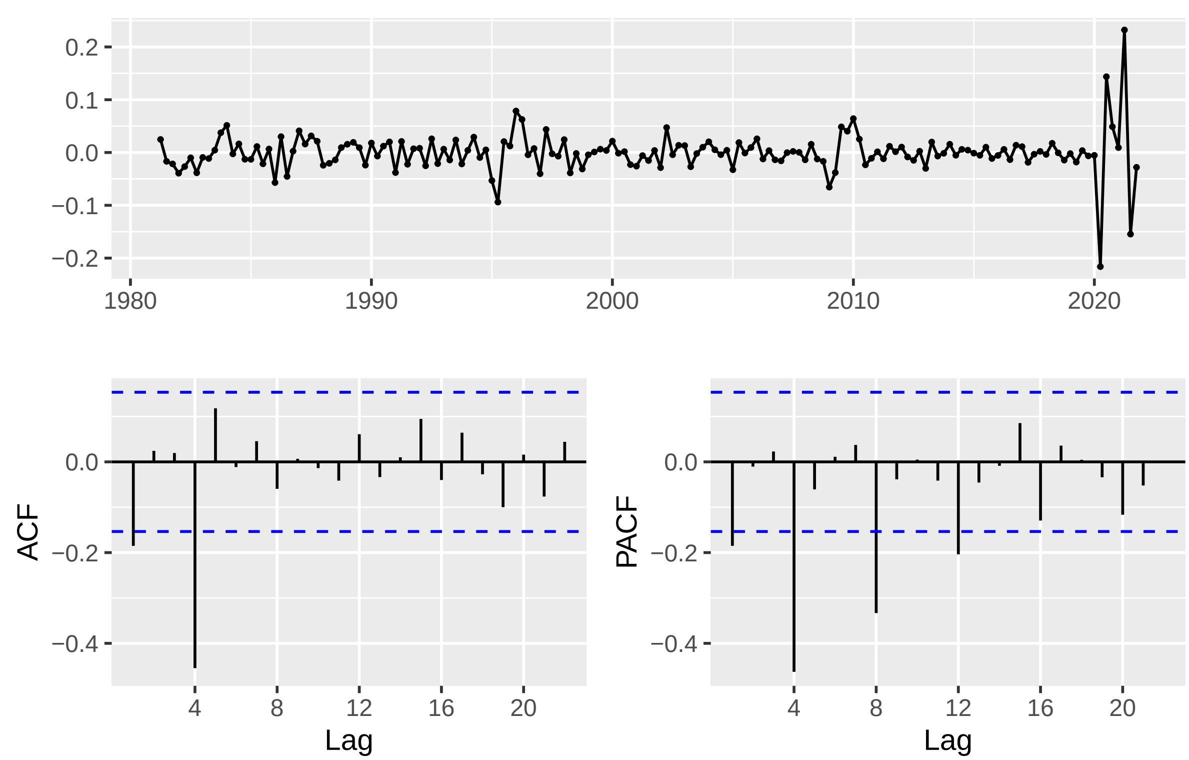


Vemos que en efecto, sí se quita la tendencia, sin embargo, seguimos viendo un componente estacional, por lo que aplicamos una diferencia estacional de periodo 4, ya que nuestra serie es trimestral.

```
log_pib_d1d4 <- diff(log_pib_d1, lag = 4)
plot(log_pib_d1d4, ylab = expression(paste(nabla[4],"(",nabla, "log(PIB)",")")))
```




```
ggtsdisplay(log_pib_d1d4)
```



Hacemos el Test Dickey-Fuller para corroborar si ahora sí, nuestra serie cumple con ser estacionaria:

```r
adf.test(log_pib_d1d4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  log_pib_d1d4
## Dickey-Fuller = -6.4061601, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

H0: No estacionaria **Ha:** Estacionaria

CONCLUSIÓN P-value= 0.01< 0.05. Rechazamos H0, por lo que nuestra serie ahora sí es estacionaria.

Identificación de los parámetros p , q , P y Q. Observemos la siguiente ACF y PACF.

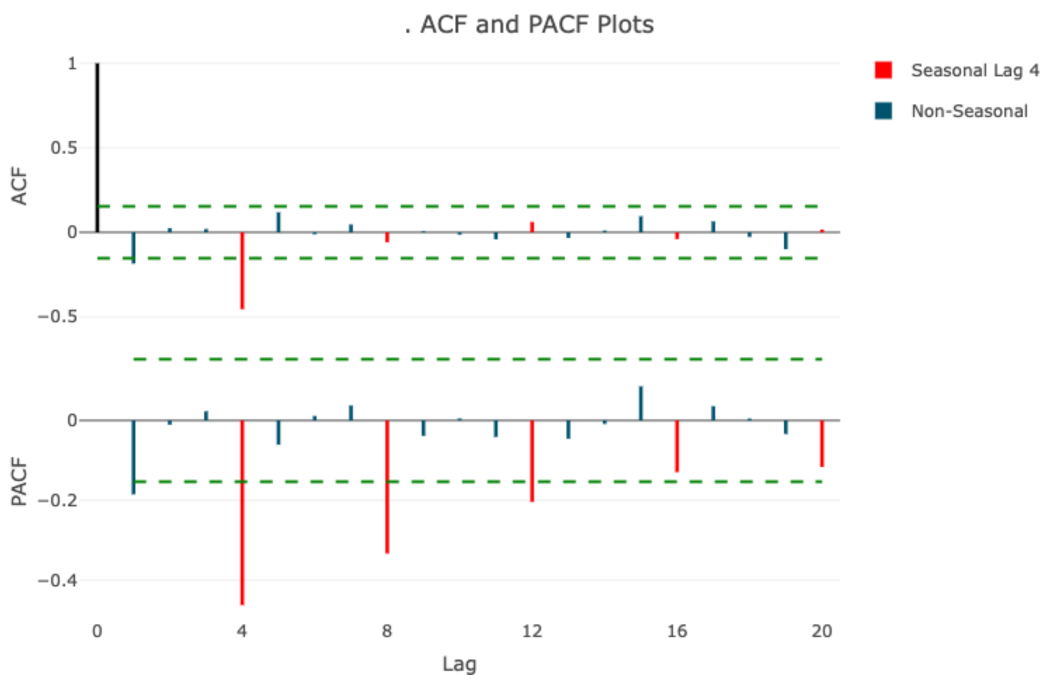


Figura 2: ACF y PACF

Con base al ACF, podemos considerar que:

- q=1 y Q=1 ***(MA)***

Y con el PACF, tenemos que:

- p=1 y P=2,3 ***(AR)***

Con esta información, haremos el análisis de los dos modelos resultantes para la estimación de los coeficientes que salen de la combinación de los parámetros anterioremente descritos

Etapa 2: Estimación

```r
modelo_1<-log_pib %>%
  Arima(order = c(1,1,1),
        seasonal = c(2,1,1))
summary(modelo_1)
```

Planteamiento de los dos modelos:

```
## Series: .
## ARIMA(1,1,1)(2,1,1)[4]
##
## Coefficients:
##              ar1        ma1        sar1        sar2        sma1
##       -0.4698439  0.2593954  -0.2619230  -0.1602728  -0.5971618
## s.e.   0.4499242  0.4950983   0.2240819   0.1896262   0.2109568
##
## sigma^2 = 0.0008511768:  log likelihood = 346.3
## AIC=-680.59   AICc=-680.06   BIC=-662.03
##
## Training set error measures:
##                            ME          RMSE           MAE             MPE
## Training set -0.001570237397 0.02829331179 0.01670300678 -0.009630491554
##                       MAPE        MASE           ACF1
## Training set 0.1006677445 0.455759307 -0.01833907106
```

```
modelo_2<-log_pib %>%
  Arima(order = c(1,1,1),
        seasonal = c(3,1,1))
summary(modelo_2)
```

```
## Series: .
## ARIMA(1,1,1)(3,1,1)[4]
##
## Coefficients:
##             ar1         ma1       sar1       sar2       sar3        sma1
##       0.7967094  -0.9643646  0.1418631  0.1419506  0.2546196  -0.9279154
## s.e.  0.0828811   0.0534297  0.1407632  0.1525085  0.1483548   0.1194216
##
## sigma^2 = 0.0008245543:  log likelihood = 348
## AIC=-682.01   AICc=-681.28   BIC=-660.35
##
## Training set error measures:
##                            ME          RMSE           MAE            MPE
## Training set -0.001744977431 0.02775906418 0.01646445397 -0.01057803331
##                        MAPE       MASE          ACF1
## Training set 0.09927867986 0.44925014 -0.1017754967
```

Vemos que tenemos un AIC más pequeño con el modelo 2, por lo que nos quedamos con dicho modelo.

b) Realiza intervalos de confianza al 95 % para los coeficientes del modelo elegido.

```
cbind(coeficientes        = modelo_2$coef,
      desviación_estandar = sqrt(diag(modelo_2$var.coef)),
      limite_inferior = modelo_2$coef - 1.96 * sqrt(diag(modelo_2$var.coef)),
      limite_superior = modelo_2$coef + 1.96 * sqrt(diag(modelo_2$var.coef)))
```

```
##       coeficientes desviación_estandar limite_inferior limite_superior
## ar1    0.7967093793       0.08288108468   0.63426245330    0.9591563052
## ma1   -0.9643645710       0.05342974124  -1.06908686382   -0.8596422782
## sar1   0.1418631253       0.14076317287  -0.13403269349    0.4177589442
## sar2   0.1419505532       0.15250845316  -0.15696601501    0.4408671214
## sar3   0.2546196351       0.14835478640  -0.03615574623    0.5453950165
## sma1  -0.9279153556       0.11942159645  -1.16198168463   -0.6938490265
```

Con base a los intervalos de confianza del 95 % del modelo 2, realizamos un modelo con P=0

```
modelo_3<-log_pib %>%
  Arima(order = c(1,1,1),
        seasonal = c(0,1,1))
summary(modelo_3)
```

```
## Series: .
## ARIMA(1,1,1)(0,1,1)[4]
##
## Coefficients:
##             ar1        ma1       sma1
##       0.7807073  -0.9569866  -0.7497620
## s.e.  0.0985825   0.0671802   0.0728452
##
## sigma^2 = 0.0008295044:  log likelihood = 346.59
## AIC=-685.18   AICc=-684.93   BIC=-672.81
##
## Training set error measures:
##                            ME          RMSE           MAE            MPE
## Training set -0.001822159812 0.02810701331 0.01642288882 -0.01104908134
##                        MAPE         MASE          ACF1
## Training set 0.09904625316 0.4481159905 -0.1114652802
```

Vemos que obtenemos un AIC aún más pequeño, realizamos los intervalos de confianza:

```
cbind(coeficientes       = modelo_3$coef,
      desviación_estandar = sqrt(diag(modelo_3$var.coef)),
      limite_inferior = modelo_3$coef - 1.96 * sqrt(diag(modelo_2$var.coef)),
      limite_superior = modelo_3$coef + 1.96 * sqrt(diag(modelo_2$var.coef)))
```

```
##       coeficientes desviación_estandar limite_inferior limite_superior
## ar1   0.7807072963       0.09858250543    0.6182603703    0.9431542222
## ma1  -0.9569866351       0.06718017205   -1.0617089279   -0.8522643422
## sar1 -0.7497619803       0.07284515914   -1.0256577991   -0.4738661614
## sar2  0.7807072963       0.09858250543    0.4817907281    1.0796238645
## sar3 -0.9569866351       0.06718017205   -1.2477620164   -0.6662112537
## sma1 -0.7497619803       0.07284515914   -0.9838283093   -0.5156956512
```

Finalmente, elegimos el modelo 3, que corresponde a un modelo SARIMA(1,1,1)x(0,1,1) $_4$

c) Realiza el diagnóstico (bondad de ajuste) de los residuales del modelo elegido para probar que es ruido blanco.

Etapa 3: Verificación de supuestos

Analizamos que los residuos sean Ruido Blanco (los residuales se distribuyen normalmente y no hay autocorrelación entre ellos).

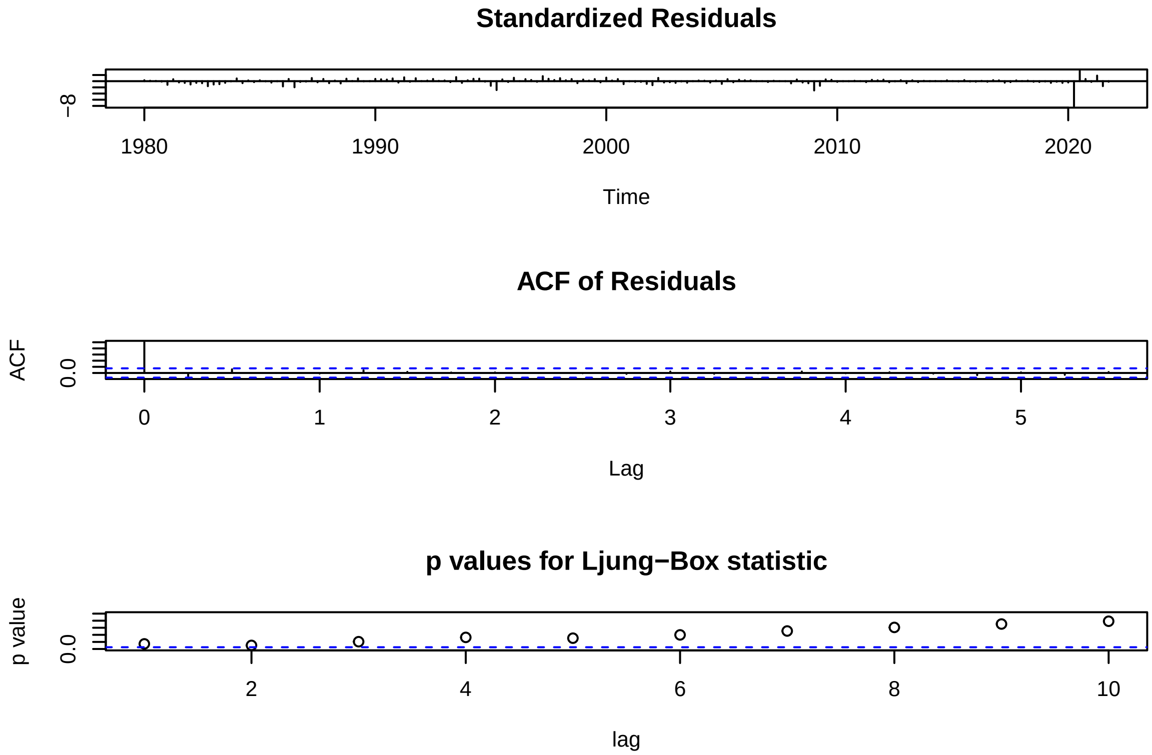
Con la prueba de Ljung-Box, se evalúa si hay o no autocorrelación en los residuales:

Hipótesis H0: No hay autocorrelación de los residuos **Ha:** Existe autocorrelación de los residuos

```
Box.test(modelo_3$residuals, lag = 20, type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  modelo_3$residuals
## X-squared = 8.7529165, df = 20, p-value = 0.9855964
```

```
par(mfrow = c(1,1))
tsdiag(modelo_3)
```

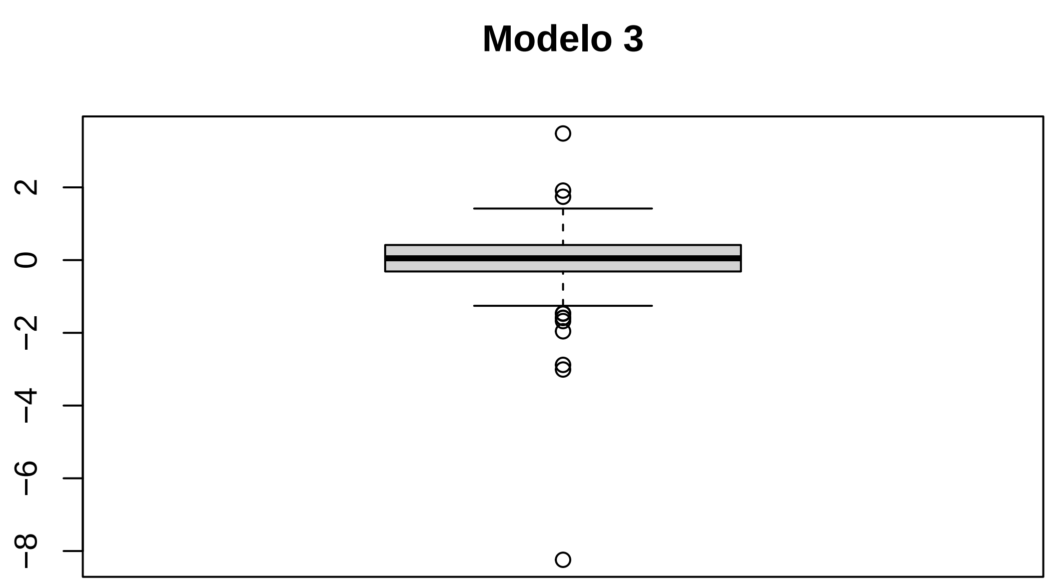


CONCLUSIÓN: Como el P-value (0.98) es mayor a 0.05 no se rechaza H0, por lo tanto, concluimos que NO hay autocorrelación de los residuos

Verificamos que los residuales tienen una distribución normal con media cero y varianza constante, para esto se realizan las siguientes pruebas de bondad de ajuste:

El Boxplot nos ayuda a visualizar la distribución de los residuales estandarizados.

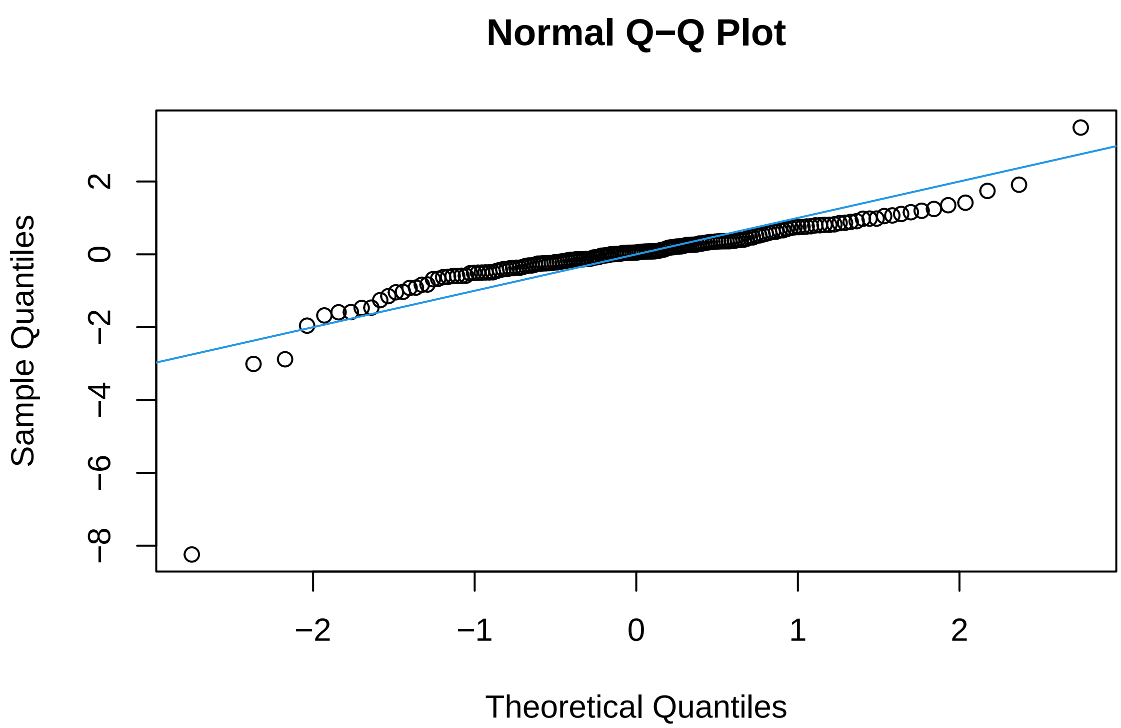
```
Z_t = (modelo_3$residuals-mean(modelo_3$residuals))/sd(modelo_3$residuals)
boxplot(Z_t, main = "Modelo 3")
```



Vemos que hay muchos outliers muy significativos, los cuales corresponden a la gran caída del PIB debido a la pandemia del COVID-19, por lo que, lo más probable es que no cumpla el supuesto de normalidad.

La siguiente gráfica nos ayuda a comparar la distribución de los residuales con base en una muestra aleatoria con distribución normal.

```
# Gráfico de QQPLOT
qqnorm(Z_t)
abline(a = 0, b = 1, col = 4)
```



Vemos que los residuales se ajustan bastante bien a una distribución normal, donde solo en los extremos se alejan de dicha distribución.

Para validar si lor residuales se distribuyen de forma normal, se aplica la Prueba de bondad de ajuste de Kolmogorov-Smirnov, que es una prueba estadística utilizada para evaluar si una muestra de datos sigue una distribución normal. Las hipótesis nula y alternativa de la prueba son las siguientes:

H0: Los residuales siguen una distribución normal. **Ha:** Los residuales NO siguen una distribución normal.

```
#Prueba de bondad de ajuste de Kolmogorov-Smirnov
ks.test(Z_t, "pnorm", 0, 1)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  Z_t
## D = 0.15297463, p-value = 0.0007695774
## alternative hypothesis: two-sided
```

Como el *p-value=0.0076* < 0.05, se puede rechazar la hipótesis nula y concluimos que los residuales no siguen una distribución normal.

La siguiente prueba se realiza para ver si la media de los residuales es igual a 0, donde:

H0: la media de los residuales es igual a 0, **Ha:** la media de los residuales es diferente de 0

```
# Prueba t para los residuales
t.test(modelo_3$residuals)
```

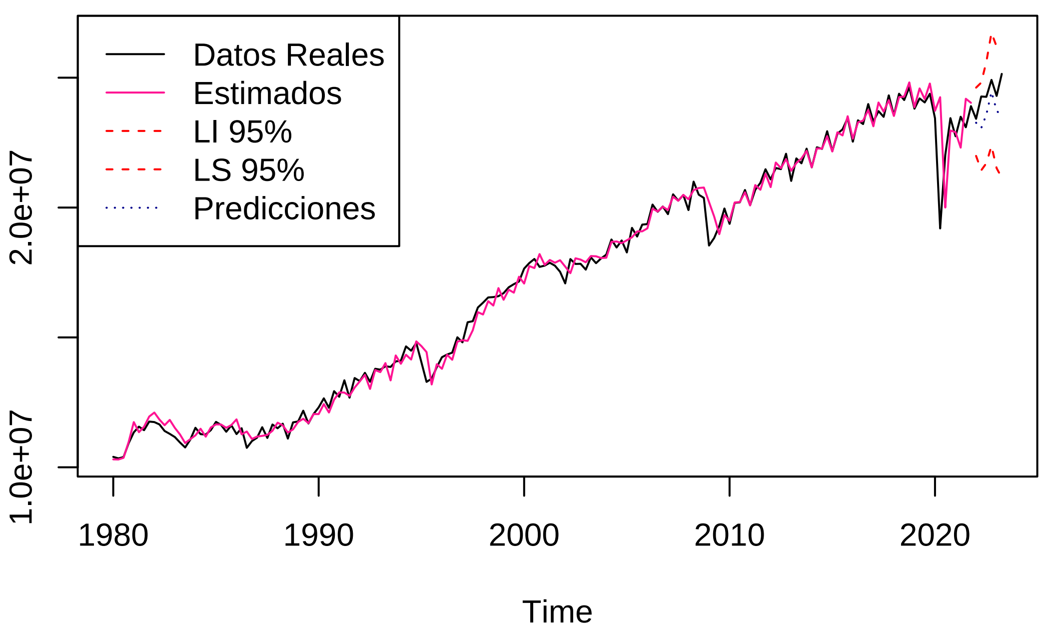
```
##
##  One Sample t-test
##
## data:  modelo_3$residuals
## t = -0.83954612, df = 167, p-value = 0.4023632
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.006107138447  0.002462818822
## sample estimates:
##       mean of x
## -0.001822159812
```

Como el *p-value=0.4023 > 0.05*, no se puede rechazar la hipótesis nula y concluimos que la media de los residuales es igual a 0.

d) En una sola gráfica muestra de la serie PBI_1 , los valores pronósticados de los 6 valores previamente eliminados, los límites inferior y superior de predicción al 95 % de confianza para valores pronósticados.

Etapa 4: Predicción

```
predicciones <- predict(modelo_3, n.ahead = 6)
estimados = log(pib)-modelo_3$residuals
li <- predicciones$pred-1.96*predicciones$se
ls <- predicciones$pred+1.96*predicciones$se
ts.plot(datos1$...2, exp(estimados), exp(li), exp(ls), exp(predicciones$pred),
lty = c(1,1,2,2,3), col = c("black","deeppink","red","red","blue4"))
legend("topleft", legend = c("Datos Reales","Estimados","LI 95%","LS 95%","Predicciones"), col = c("black","deeppi
       lty = c(1,1,2,2,3))
```



e) Elabora una tabla que muestre: los errores entre el pronóstico y los valores reales que se eliminaron y verifica en la tabla, si el valor real se encuentra dentro del intervalo de confianza al 95 %.

```
Tabla <- data.frame("Valores Reales"=datos2[c(169:174)],
                    "Pronósticos"=exp(predicciones$pred),
```

```r
                    "Error Relativo"=abs((datos2[c(169:174)]-exp(predicciones$pred))
                                         /datos2[c(169:174)]),
                    "Error absoluto"=abs(datos2[c(169:174)]-exp(predicciones$pred)),
                    "Límites Inferiores"=exp(li),
                    "Límites Superiores"=exp(ls))
kable(Tabla, align = "c", caption = "Resumen general") %>%
  kable_styling(latex_options = "HOLD_position")
```

Cuadro 1: Resumen general

Valores.Reales	Pronósticos	Error.Relativo	Error.absoluto	Límites.Inferiores	Límites.Superiores
23416073.47	23265563.59	0.006427631236	150509.8853	21988599.40	24616686.09
24276153.48	23059224.28	0.050128584095	1216929.2015	21432969.62	24808873.15
24263053.29	23580411.13	0.028135047624	682642.1599	21707516.40	25614896.65
24916415.11	24452958.41	0.018600456508	463456.6956	22371326.51	26728284.30
24298924.43	23739878.91	0.023007006919	559045.5225	21514906.81	26194947.33
25146771.16	23483458.95	0.066144166308	1663312.2136	21148159.88	26076634.90

Podemos notar que los valores reales se encuentran dentro del intervalo de confianza al 95 %, por lo que podemos concluir, que nuestro modelo predice de forma satisfactoria.