Regresión lineal con series de tiempo.

Amelia Cristina Herrera Briceño, melinnicri@gmail.com

Precios IPC.

Datos de IPC, Índice de Precios al Consumidor, Lima, Perú; mide la variación promedio de los precios de una canasta de bienes y servicios representativa del consumo de los hogares. Aunque se calcula a nivel nacional, también se publica específicamente para Lima Metropolitana, que es una de las áreas más importantes para el análisis económico en el país.

Tipos de Datos: fecha (mes-año, caracteres de tiempo) desde 1992 al 2023 y precio IPC (numérico, continuo).

1. **Análisis de Descomposición:** Para la aplicación del Método Aditivo o Método Multiplicativo, se escoge el M. Multiplicativo porque la curva en general es ascendente con un par de picos también ascendentes y varianza cambia en con el tiempo.

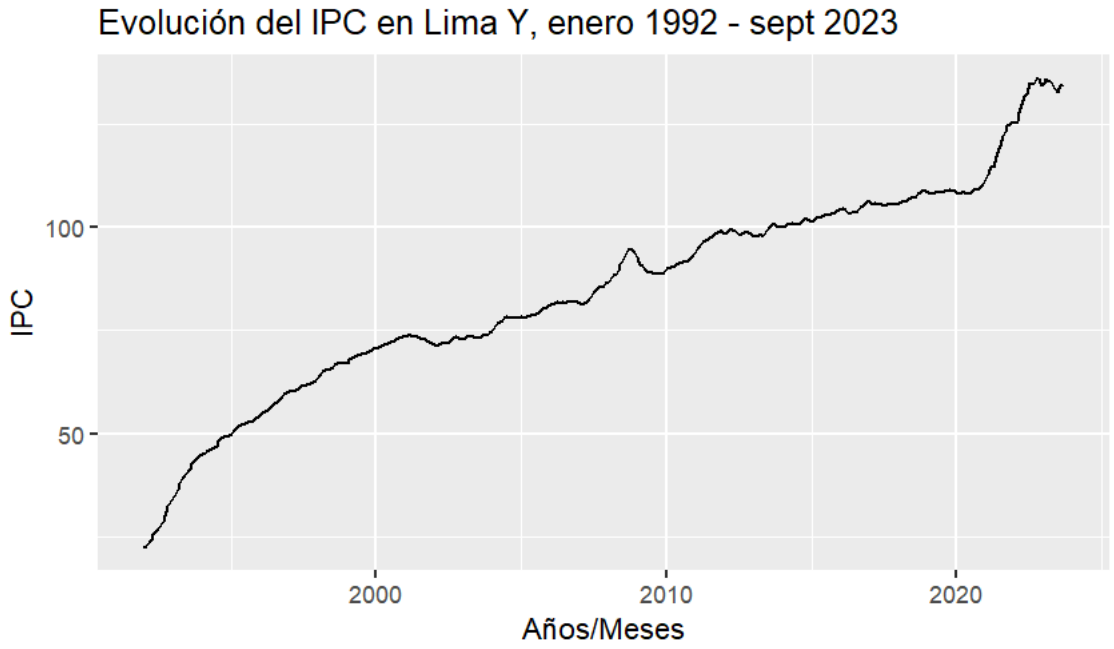
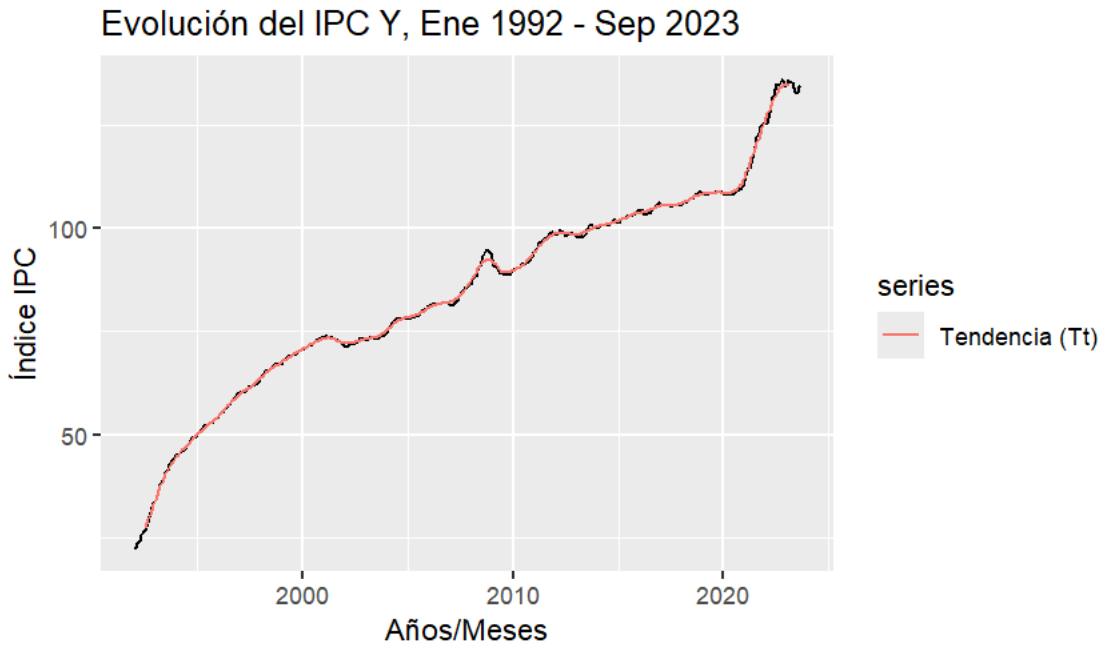


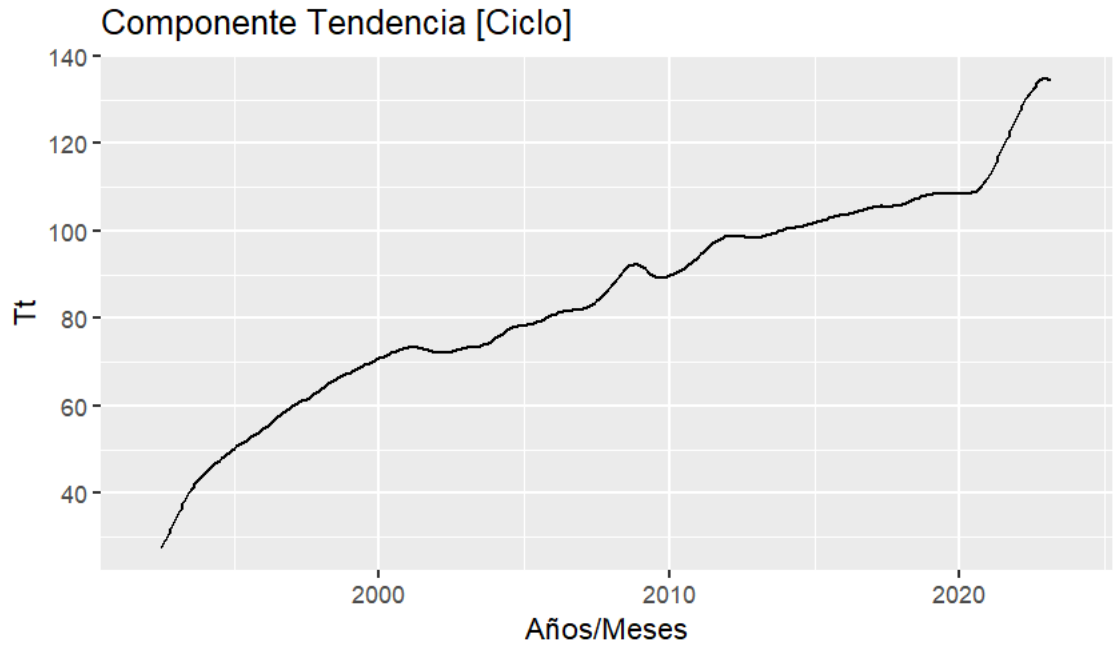
Gráfico de la Evolución del IPC durante los años 1992 al 2023.

Es una evolución ascendente en el tiempo de los Índices de Precios al Consumidor.

De acuerdo con el Modo Aditivo, su **Tendencia** sería ascendente, y a pesar de ambos picos eventuales ascendentes, continúa hacia el alza, de acuerdo con el siguiente gráfico de tendencia (Tt):

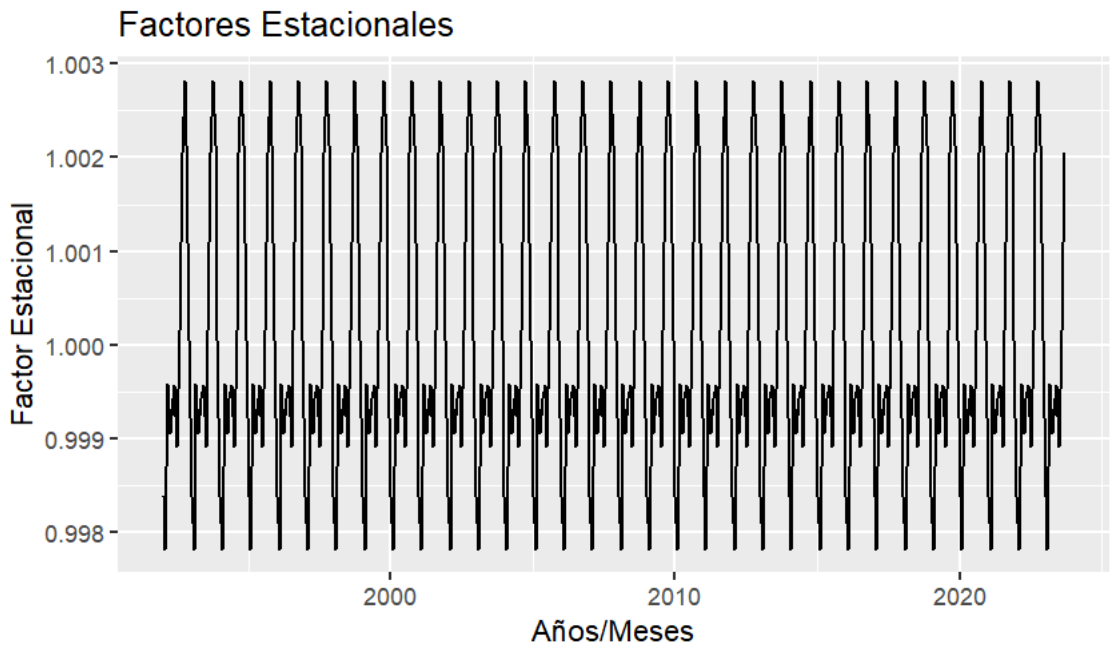


De acuerdo con el modo Multiplicativo, El **Componente Tendencia del ciclo**, como se puede apreciar en el siguiente gráfico:



Va de 40 a 140, significa que ha tendido al aumento en promedio de precios de bienes y servicios durante los años 1992 al 2023, en Lima. Al mismo tiempo el valor adquisitivo del dinero disminuye porque ya no se pueden adquirir con la misma cantidad de dinero cierta

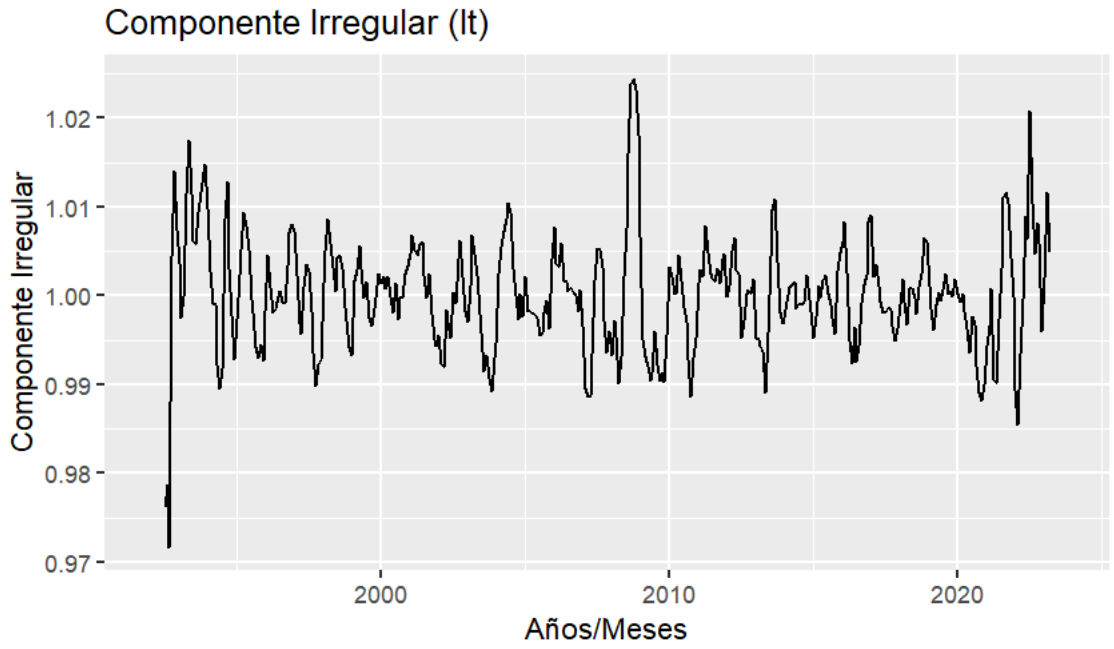
cantidad de bienes y servicios en el pasado, por lo que se deben ajustar los sueldos e ingresos de las personas periódicamente para mantener el poder adquisitivo de éstas.



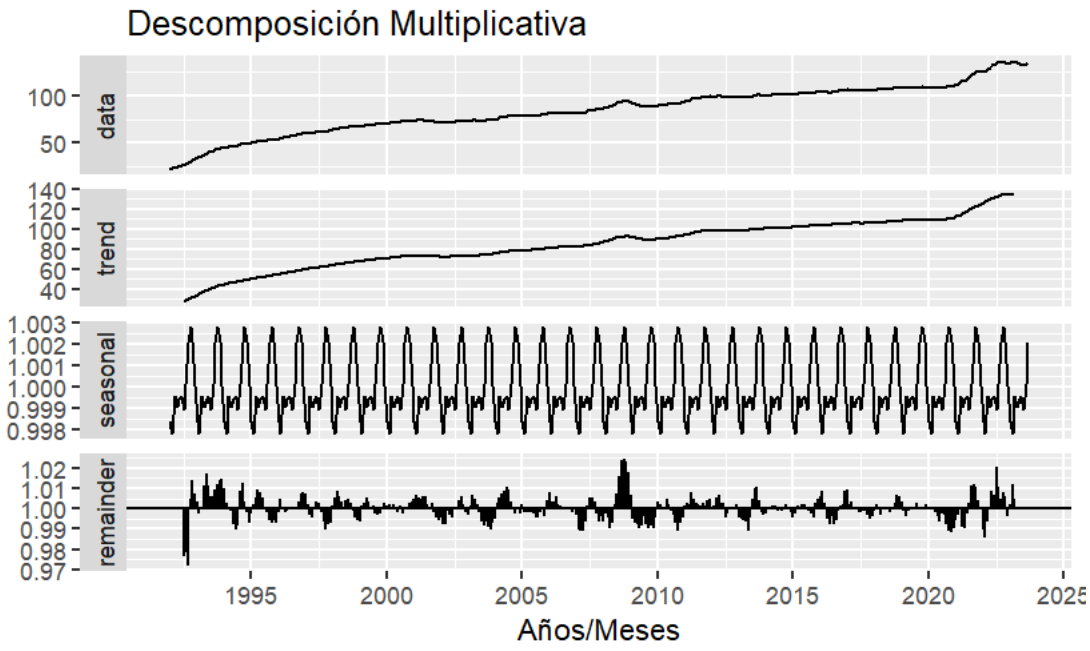
Factores estacionales: existiría una **Baja estacionalidad:** Los factores estacionales están muy cercanos a 1 (no hay sesgo estacional significativo), lo que indica una variación estacional muy pequeña. Esto significa que los precios en general no presentan fluctuaciones significativas a lo largo del año.

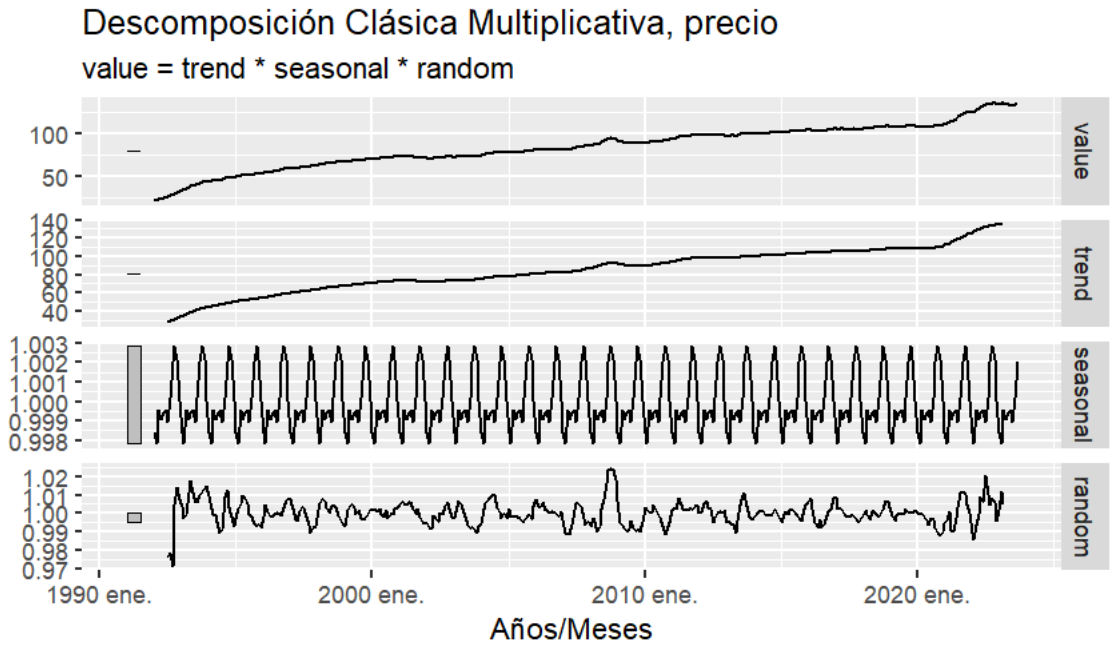
Tabla "Factores Estacionales":

1	2	3	4	5	6
0.9983957	0.9978241	0.9995862	0.9990637	0.9994232	0.9995755
7	8	9	10	11	12
0.9989145	1.0001529	1.0020497	1.0028185	1.0021352	1.0000607



En cuanto al **Componente Irregular (It)**, que no se puede explicar con la tendencia, la estacionalidad o el ciclo, estos eventos inesperados (los picos en alza), pueden ser choques económicos, crisis, cambios de políticas gubernamentales, incluso conflictos bélicos o desastres naturales. 2008-2009: Crisis económica-financiera mundial, el fenómeno del Niño costero en Lima, sequía, impacto agropecuario negativo; 2020-2021: Pandemia del Covid-19. 2022: Conflicto de Ucrania - Rusia, con impacto en los precios de combustibles y productos.





Acá en estas dos gráficas, de **Descomposición Multiplicativa y la forma clásica (Value = trend * seasonal * random)**, se pueden ver que por separado son muy parecidas las curvas obtenidas en este resumen.

Código en R Studio de la Descomposición de la serie de tiempo de los datos de IPC, Lima, Perú.

```
# Análisis de descomposición:

# Instalar y cargar los paquetes necesarios
packages <- c("haven", "forecast", "ggplot2", "dplyr", "magrittr")
new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
if (length(new_packages)) install.packages(new_packages)

lapply(packages, library, character.only = TRUE)


# Cargar el paquete
library(readr)

# Leer el archivo CSV como serie
serie.precio <-
read.csv("C:/Users/56988/Desktop/ClasesDS/Series_tiempo_Rstudio/sesion05/arreglovariables/ipc.csv",
sep = ",", dec = ".", encoding = "Latin1")

# fecha y PN38712PM
serie.precio

# Verificar la estructura de los datos
str(serie.precio)

# Mostrar las primeras filas de los datos
head(serie.precio)

# Mostrar las últimas filas
tail(serie.precio)

# Crear la serie temporal (reemplaza con tus datos si es necesario)
serie.precio.ts <- ts(serie.precio$PN38712PM, start = c(1992, 1), frequency = 12)

# Verificar la serie temporal
print(serie.precio.ts)

# Graficar la serie temporal
plot(serie.precio.ts)

install.packages("forecast")
install.packages("ggplot2")

library(forecast)
library(ggplot2)


# Graficamos la serie temporal
autoplot(serie.precio.ts) +
  ggtitle("Evolución del IPC Y, Ene 1992 - Sep 2023") +
  xlab("Años/Meses") +
```

```
  ylab("Índice IPC")

####################
## Modelo Aditivo ##
####################

# 1.1. Componente de Tendencia Tt [Componente TCt]
ma2_12 <- ma(serie.precio.ts, order = 12, centre = TRUE)
autoplot(serie.precio.ts) +
  ggtitle("Evolución del IPC Y, Ene 1992 - Sep 2023") +
  xlab("Años/Meses") +
  ylab("Índice IPC") +
  autolayer(ma2_12, series = "Tendencia (Tt)")

# 1.2. Cálculos de los Factores Estacionales [Componente St]
# Calcular la serie ajustada (SI) eliminando la tendencia
Yt <- serie.precio.ts  # serie original
Tt <- ma2_12          # media móvil centrada
SI <- Yt - Tt        # Diferencia: componente estacional e irregular

# Calcular el componente estacional (St) como el promedio de los residuos
St <- tapply(SI, cycle(SI), mean, na.rm = TRUE)

# Ajustar los factores estacionales para que sumen 0
St <- St - mean(St)

# Mostrar los factores estacionales en consola
print("Factores Estacionales:")
print(St)

# Generar la serie de factores estacionales ajustados para la serie temporal completa
St_series_adjusted <- rep(St, length.out = length(Yt)) %>% ts(start = start(serie.precio.ts), frequency = 12)

# Graficar los factores estacionales ajustados
autoplot(St_series_adjusted) +
  ggtitle("Factores Estacionales Ajustados") +
  xlab("Años/Meses") +
  ylab("Factor Estacional")


# 1.3. Cálculo del Componente Irregular [It]; It = Yt - Tt - St
It <- Yt - Tt - St_series_adjusted

# Graficar el componente irregular
autoplot(It) +
  ggtitle("Componente Irregular (It)") +
  xlab("Años/Meses") +
  ylab("Componente Irregular")


# 1.4. Descomposicion Aditiva (usando la libreria Stats)

descomposicion_aditiva<-decompose(serie.precio.ts,type = "additive")
```

```
autoplot(descomposicion_aditiva,main="Descomposicion Aditiva",
       xlab="Años/Meses")

# 1.5. Descomposicion Aditiva (usando libreria feasts)

library(tsibble)
library(feasts)
library(ggplot2)
Yt%>%as_tsibble()%>% #convertir serie temporal a tsibble
  model(           #convertir a modelo de descomposicion clasica
    classical_decomposition(value,type = "additive")
  )%>%
  components()%>%   #pedimos que extraiga los componentes
  autoplot() +      #pedimos que grafique
  labs(title = "Descomposicion Clasica Aditiva, precio")+
  xlab("Años/Meses")


############################
## Modelo Multiplicativo ##
############################

# 2.1. Componente Tendencia ciclo [Tt=TCt]

Tt<- ma(serie.precio.ts, 12, centre = TRUE)
autoplot(Tt,main = "Componente Tendencia [Ciclo]", xlab = "Años/Meses",ylab = "Tt")

# 2.2. Calculos de Factores Estacionales [St]

SI<-Yt/Tt               #Serie sin tendencia.

# Calcular los factores estacionales (St) promediando los resultados de cada mes
St <- tapply(SI, cycle(SI), mean, na.rm = TRUE)

# Ajustar los factores estacionales para que sumen "1" en el modelo multiplicativo
St <- St * 12 / sum(St)

# Generar la serie de factores estacionales para cada valor de la serie original
St_series <- rep(St, length.out = length(Yt)) %>% ts(start = start(serie.precio.ts), frequency = 12)

# Mostrar los factores estacionales en consola
print("Factores Estacionales:")
print(St)

# Graficar los factores estacionales
autoplot(St_series) +
  ggtitle("Factores Estacionales") +
  xlab("Años/Meses") +
  ylab("Factor Estacional")


# 2.3. Calculo del componente Irregular [It]
```

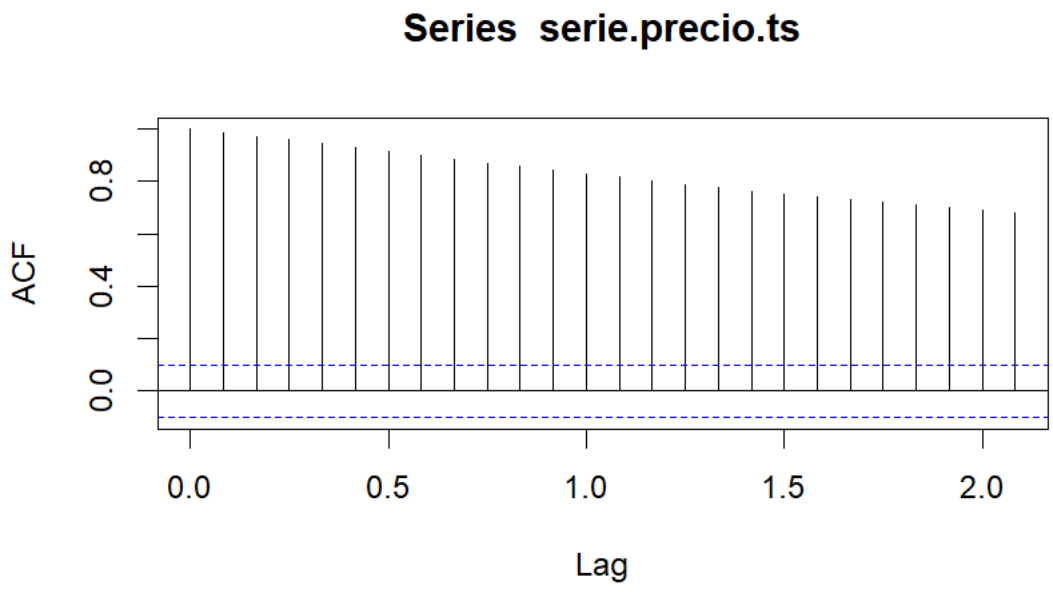
```
It <- Yt / (Tt * St_series)

# Graficar el componente irregular
autoplot(It) +
  ggtitle("Componente Irregular (It)") +
  xlab("Años/Meses") +
  ylab("Componente Irregular")


# 2.4. Descomposicion multiplicativa (usando libreria stats)

descomposicion_multiplicatica<-decompose(serie.precio.ts,type = "multiplicative")
autoplot(descomposicion_multiplicatica,main="Descomposición Multiplicativa",xlab="Años/Meses")


# 2.5. Descomposicion multiplicativa (usando libreria feasts)

library(tsibble)
library(feasts)
library(ggplot2)
Yt %>% as_tsibble() %>%
  model(classical_decomposition(value, type = "multiplicative")) %>%
  components() %>%
  autoplot() +
  labs(title = "Descomposición Clásica Multiplicativa, precio") + xlab("Años/Meses")


#########
# Descomposicion (usando la libreria TSstudio)
#########

# Intenta instalar TSstudio desde CRAN
install.packages("TSstudio", repos = "https://cloud.r-project.org/")

# Verifica que el paquete esté instalado
if ("TSstudio" %in% rownames(installed.packages())) {
  # Cargar el paquete
  library(TSstudio)
} else {
  print("El paquete TSstudio no se pudo instalar. Verifica los errores de instalación.")
}

library(TSstudio)
# Aditivo
ts_decompose(Yt, type = "additive", showline = TRUE)
# Multiplicativo
ts_decompose(Yt, type = "multiplicative", showline = TRUE)
```

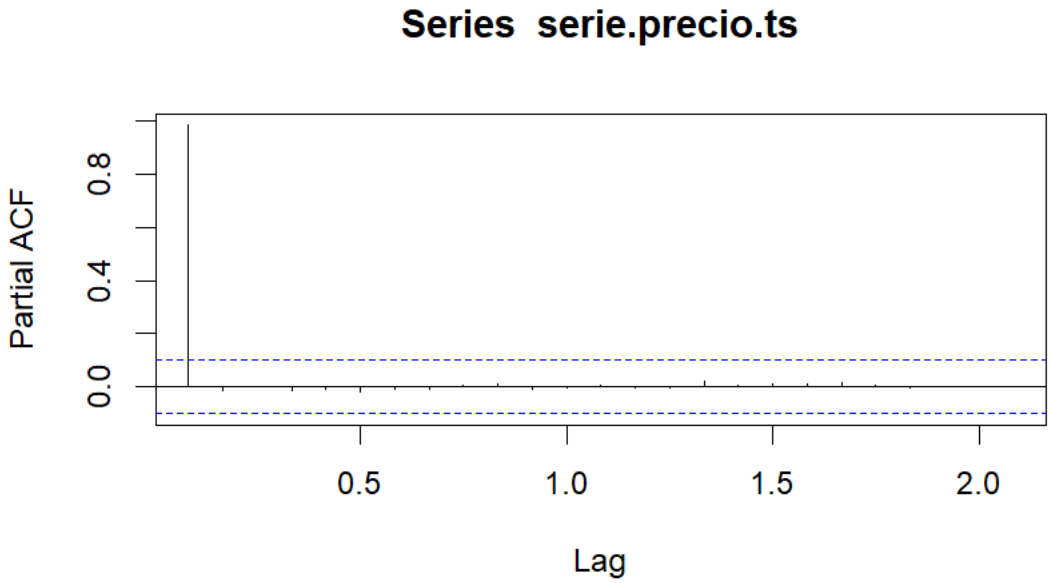
2. **Test de Raíz unitaria:** presencia de tendencia estocástica de los datos de serie de tiempo 1992-2023, IPC de Lima, Perú, probando robustez, heterocedasticidad y autocorrelación, para generar pronósticos u otros análisis.

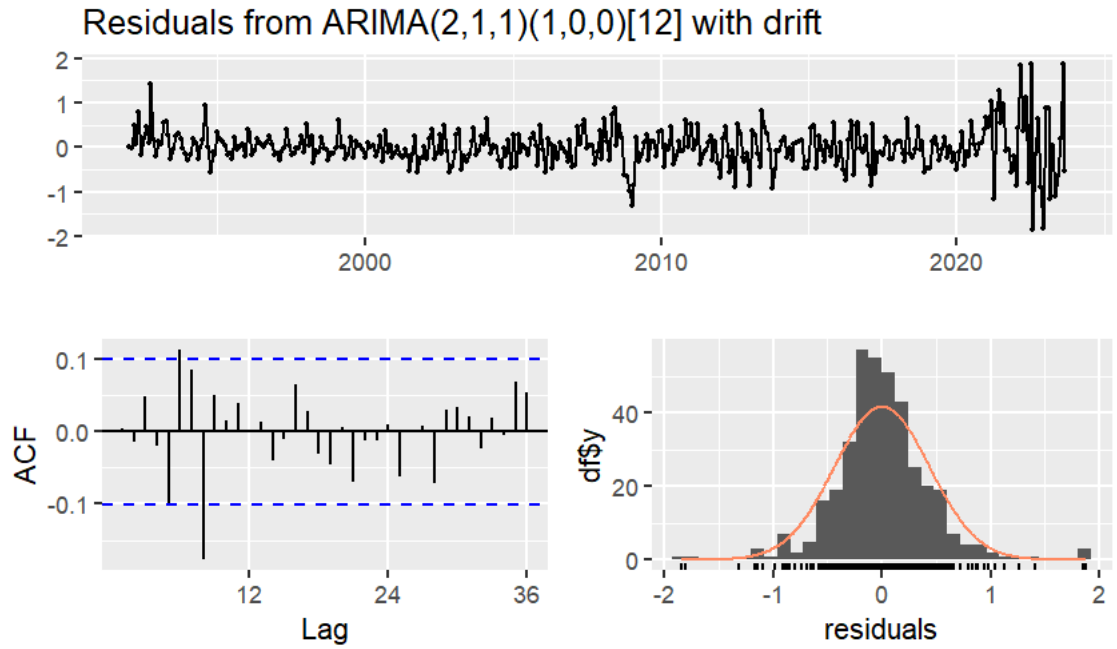
- Con la Prueba de raíz unitaria, **Prueba de Dickey-Fuller Aumentada (ADF)**: serie estacionaria.
- Con la **Prueba de Phillips-Perron (PP)**: serie estacionaria.
- Con la **Prueba de Kwiatkowski-Phillips-Schmidt-Shin (KPSS)**: serie no estacionaria.
- Con la **Prueba de Zivot-Andrews**: serie estacionaria con ruptura (posición 353).

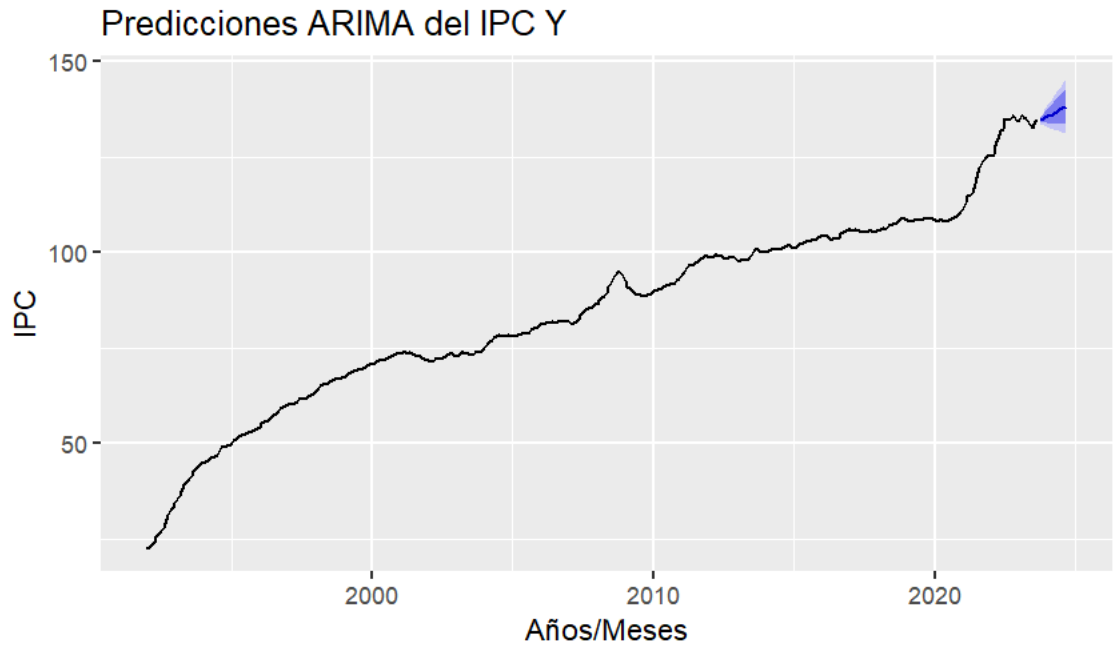
3. Con la aplicación del **modelo ARIMA:**

- **Modelo ARIMA (2, 1, 1)(1,0,0)[12] con deriva, con Coeficientes**:
  - **ar1**: 1.0835 (coeficiente del primer término autorregresivo)
  - **ar2**: -0.1868 (coeficiente del segundo término autorregresivo)
  - **ma1**: -0.6416 (coeficiente del primer término de media móvil)
  - **sar1**: -0.0796 (coeficiente del término autorregresivo estacional)
  - **drift**: 0.2971 (término de deriva)




Series serie.precio.ts
Partial ACF
0.0
0.4
0.8
0.5
1.0
1.5
2.0
Lag


Residuals from ARIMA(2,1,1)(1,0,0)[12] with drift
2
1
0
-1
-2
2000
2010
2020
ACF
0.1
0.0
-0.1
12
24
36
Lag
df$y
40
20
0
-2
-1
0
1
2
residuals



Los errores estándar (s.e.) de estos coeficientes también se proporcionan, lo que indica la precisión de las estimaciones.

- **sigma^2**: 0.1954
  - Esto representa la varianza del error residual del modelo.
- **Log likelihood**: -226.7
  - El logaritmo de la verosimilitud del modelo, que se utiliza para comparar modelos.
- **AIC (Akaike Information Criterion)**: 465.4
  - **AICc**: 465.62 (AIC corregido para pequeñas muestras)
  - **BIC (Bayesian Information Criterion)**: 489.04
  - Estos criterios se utilizan para evaluar la calidad del modelo, con valores más bajos indicando un mejor ajuste.

Medidas de error del conjunto de entrenamiento:

1. **ME (Mean Error)**: -0.0008747821
   - Error medio de las predicciones.
2. **RMSE (Root Mean Squared Error)**: 0.4385175
   - Raíz del error cuadrático medio, una medida de la precisión de las predicciones.
3. **MAE (Mean Absolute Error)**: 0.3098639
   - Error absoluto medio, otra medida de la precisión de las predicciones.
4. **MPE (Mean Percentage Error)**: 0.02761451
   - Error porcentual medio.
5. **MAPE (Mean Absolute Percentage Error)**: 0.3766783
   - Error porcentual absoluto medio.
6. **MASE (Mean Absolute Scaled Error)**: 0.08062163
   - Error absoluto medio escalado.
7. **ACF1**: 0.003244953
   - Autocorrelación del primer rezago de los residuos.

Por lo tanto, los coeficientes tienen errores estándar relativamente bajos, lo que sugiere que son estimaciones precisas, significativas.
Las medidas de error, mientras más bajos, mejores.
AIC, AICc y BIC son bajos, lo que indican un mejor ajuste.
Autocorrelación de residuos, ACF1 cercano a 0 sugiere que los residuos no están autocorrelacionados, lo que es deseable.

En cuanto al **test de Ljung-Box** se utiliza para verificar **si hay autocorrelación en los residuos** de un modelo de series temporales, como un modelo ARIMA.
Resultados del Test de Ljung-Box:

- *Q = 33.016**: Este es el estadístico de prueba de Ljung-Box.
- **df = 20**: Grados de libertad, que en este caso es el número de rezagos utilizados en el test menos el número de parámetros estimados en el modelo.
- **p-value = 0.0336**: El p-valor asociado con el estadístico de prueba.

Interpretación:

1. **Hipótesis Nula (H0)**: Los residuos no están autocorrelacionados (es decir, son ruido blanco).
2. **Hipótesis Alternativa (H1)**: Los residuos están autocorrelacionados.

Para interpretar el p-valor:

- **Si el p-valor es menor que el nivel de significancia (por ejemplo, 0.05)**, se rechaza la hipótesis nula. Esto sugiere que hay autocorrelación en los residuos.
- **Si el p-valor es mayor que el nivel de significancia**, no se rechaza la hipótesis nula, lo que sugiere que los residuos no están autocorrelacionados.

Por lo tanto, el p-valor es **0.0336**, que es menor que 0.05. Esto significa que se puede **rechazar la hipótesis nula** y concluir que hay **autocorrelación en los residuos** del modelo ARIMA(2,1,1)(1,0,0)[12] con deriva.

Implicaciones:
La presencia de autocorrelación en los residuos indica que el modelo ARIMA no está capturando toda la estructura de la serie temporal. Esto sugiere que se podría necesitar:

- **Revisar el modelo**: Considerar un modelo ARIMA diferente o ajustar los parámetros.
- **Incluir términos adicionales**: Como términos estacionales o de tendencia.
- **Transformar los datos**: Aplicar diferenciación adicional o eliminar tendencias.

***Se intentó separar el modelo ARIMA, pero no hubo grandes diferencias.**

En la Predicción final, como se puede ver en el último gráfico, la ruptura ocasionada por la pandemia del covid-19, afectó fuertemente los Índices de Precios al Consumidor en Lima, Perú, esperando una nueva tendencia al alza por lo pronto.

Código RStudio adjunto para Raíz unitaria y Modelo ARIMA:

```
# Instalar y cargar los paquetes necesarios
packages <- c("haven", "forecast", "ggplot2", "dplyr", "magrittr", "TTR", "tseries", "urca")
new_packages <- packages[!(packages %in% installed.packages()[,"Package"])]
if (length(new_packages)) install.packages(new_packages)
lapply(packages, library, character.only = TRUE)

# Instalar el paquete (si no está instalado)
install.packages("readr")

# Cargar el paquete
library(readr)

# Leer el archivo CSV como serie
serie.precio <-
read.csv("C:/Users/56988/Desktop/ClasesDS/Series_tiempo_Rstudio/sesion05/arreglovariables/ipc.csv")

# Revisar datos
head(serie.precio)
tail(serie.precio)

# Generar la Serie Temporal
serie.precio.ts <- ts(serie.precio$PN38712PM, start = c(1992, 1), frequency = 12)

# Graficar la serie temporal
autoplot(serie.precio.ts) +
  ggtitle("Evolución del IPC en Lima Y, enero 1992 - sept 2023") +
  xlab("Años/Meses") +
  ylab("IPC")

# Realizar pruebas de raíz unitaria
# Prueba de Dickey-Fuller Aumentada (ADF)
adf_test <- adf.test(serie.precio.ts, alternative="stationary")
print(adf_test)

# H0: La serie tiene una raíz unitaria (no es estacionaria);
# H1: La serie no tiene una raíz unitaria (es estacionaria);
# si -3.6673 es menor que -2,8699, H0 se rechaza; si -3.6673 es mayor
# que -2,8699, H0 no se rechaza; por lo tanto, la serie no tiene una
# raíz unitaria, lo cual es estacionaria.

# Estadístico de prueba: -3.6673
# Valor p (probabilidad): 0.02669 *el p-valor te dice cuán probable
# es observar los datos que tienes si la serie realmente tiene
# una raíz unitaria (es decir, no es estacionaria)
# nivel de significancia: 0,05 (-2.8699298018856574)
# Lag order = 7, significa las 7 veces de resagos de la serie.

# Prueba de Phillips-Perron (PP)
# Nota: La función adf.test realiza la prueba ADF y PP en la versión moderna de tseries
```

```
pp_test <- adf.test(serie.precio.ts, alternative="stationary", k=0)  # k=0 para PP
print(pp_test)

# H0: La serie tiene una raíz unitaria (no es estacionaria);
# H1: La serie no tiene una raíz unitaria (es estacionaria);
# si -4.2171 es menor que -3,45, H0 se rechaza; si -4.2171 es mayor
# que -3,45, H0 no se rechaza; por lo tanto, la serie no tiene una
# raíz unitaria, lo cual es estacionaria.
# Para 5% de significancia: -2,87
# Para 1% de significancia: -3,45

# Prueba de Kwiatkowski-Phillips-Schmidt-Shin (KPSS)
kpss_test <- ur.kpss(serie.precio.ts)
summary(kpss_test)


# Critical value for a significance level of:
#  10pct  5pct 2.5pct  1pct
# critical values 0.347 0.463  0.574 0.739

# Hipótesis nula (H0): La serie es estacionaria alrededor
# de una tendencia o media.
# Hipótesis alternativa (H1): La serie tiene una raíz unitaria
# (es no estacionaria).
# Si 6.0145 es menor que los valores críticos al 5%,
# H0 no se rechaza; si 6.0145 es mayor
# que valores críticos, H0 se rechaza; por lo tanto,
# la serie es no estacionaria.


# Test de Zivot-Andrews
za_test <- ur.za(serie.precio.ts, model = "both", lag = 0)
summary(za_test)

# Según esta prueba, la serie de tiempo es estacionaria con una ruptura
# en la posición 353 (por eso la contradicción de las 3 pruebas anteriores).

# Ajustar el modelo ARIMA si la serie es estacionaria
# En caso contrario, realizar diferenciación
if (adf_test$p.value > 0.05) {
  serie.precio.ts_diff <- diff(serie.precio.ts)
  adf_test_diff <- adf.test(serie.precio.ts_diff, alternative="stationary")
  print(adf_test_diff)

  # Ajustar el modelo ARIMA en la serie diferenciada
  arima_model <- auto.arima(serie.precio.ts_diff)
} else {
  # Ajustar el modelo ARIMA en la serie original si ya es estacionaria
  arima_model <- auto.arima(serie.precio.ts)
}

# Resumen del modelo ARIMA
summary(arima_model)
```

```
# Correlogramas simple y parcial
acf(serie.precio.ts)
pacf(serie.precio.ts)

# Aquí no se realizó la diferenciación, porque no se cumplió en el
# modelo de la serie diferenciada

# Correlogramas simple y parcial de la serie diferenciada
acf(serie.precio.ts_diff)
pacf(serie.precio.ts_diff)


# Diagnóstico del modelo
checkresiduals(arima_model)

# Predicciones
forecast_arima <- forecast(arima_model, h = 12)  # Predecir los próximos 12 períodos

# Graficar las predicciones
autoplot(forecast_arima) +
  ggtitle("Predicciones ARIMA del IPC Y") +
  xlab("Años/Meses") +
  ylab("IPC")



# Intento de ajuste nuevo, separando el modelo ARIMA
# Ajustar diferentes modelos ARIMA
arima_model1 <- auto.arima(serie.precio.ts, seasonal=FALSE)
arima_model2 <- auto.arima(serie.precio.ts, seasonal=TRUE)

# Comparar AIC y BIC
AIC(arima_model1, arima_model2)
BIC(arima_model1, arima_model2)

# Diagnóstico de residuos
checkresiduals(arima_model1)
checkresiduals(arima_model2)

# Seleccionar el mejor modelo basado en AIC, BIC y diagnósticos de residuos
best_model <- ifelse(AIC(arima_model1) < AIC(arima_model2), arima_model1, arima_model2)

# Predicciones con el mejor modelo
forecast_arima <- forecast(best_model, h=12)
autoplot(forecast_arima) +
  ggtitle("Predicciones ARIMA del IPC Y") +
  xlab("Años/Meses") +
  ylab("IPC")
```