La población carcelaria en Colombia 1991-2017

Sergio Solano

Febrero de 2016

Data

El INPEC publica mensualemente la serie población carcelaria, desde 1991 hasta el mese anterior a la publicación. Esta serie se encuentra separada por situación jurídica (condenados, sindicados) y por genero.

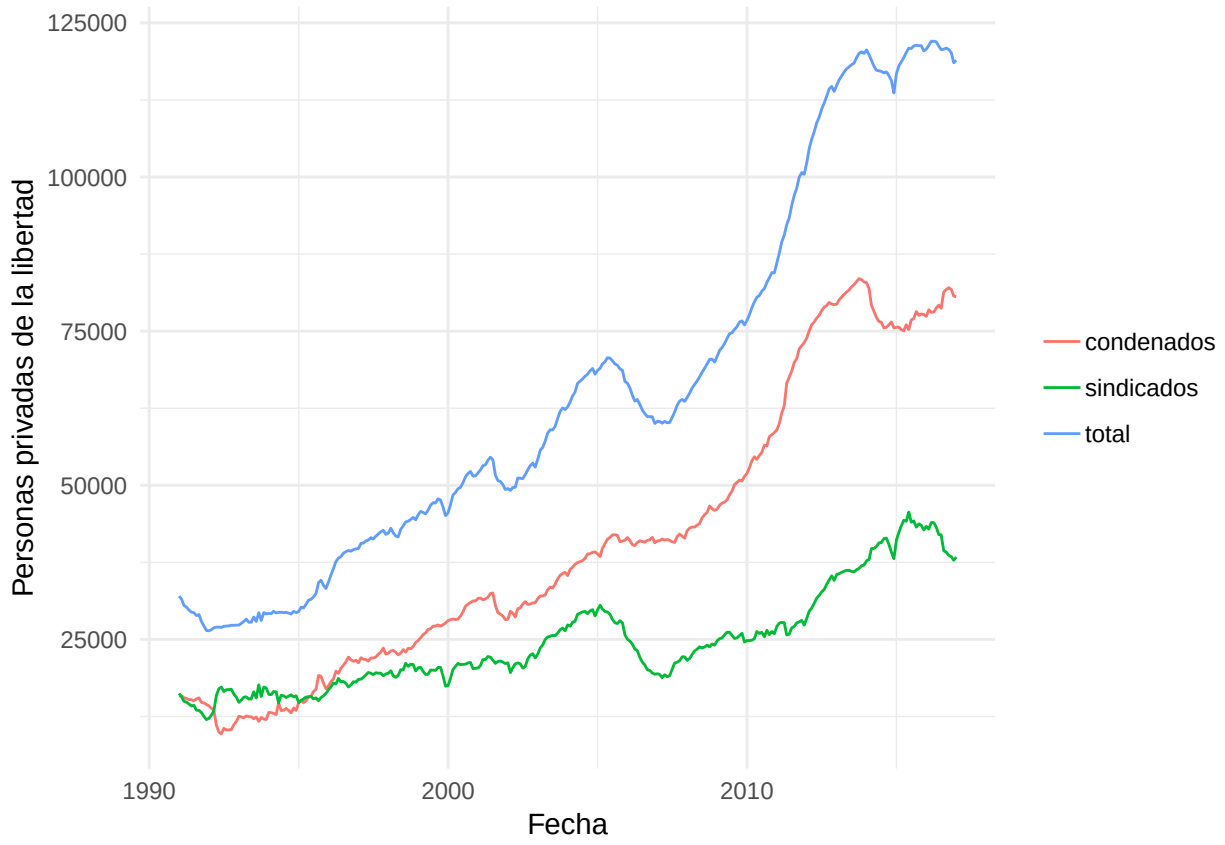
Objetivo

El proceso a través del cual las personas pasan de una situación jurídica a otra es conocido, y sin embargo no observado, pues no se publican las series de tiempo que reflejan esta transición (cantidad de personas sentenciadas por mes, cantidad de personas liberadas mes, duración de la condena).

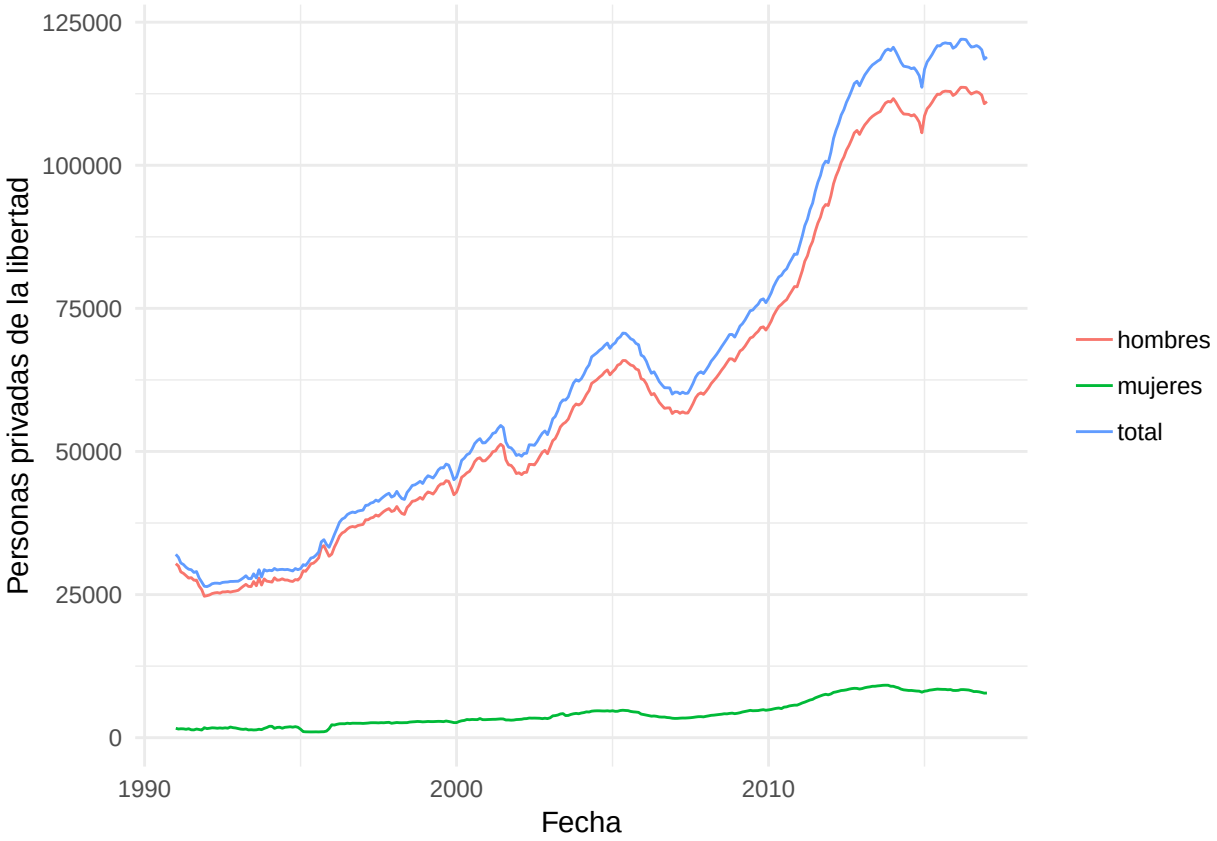
Este ejercicio de demografía, enmarcado en el estudio de poblaciones pequeñas, presenta la oportunidad de comparar la efectividad de diferentes métodos de proyección.

Análisis exploratorio

La población carcelaria total entre 1991 y 2017 se ha cuadruplicado, al pasar de 32.036 a 128.125 internos. Aunque la mayoría son hombres la población carcelaria feminina en el mismo periodo ha crecido a una tasa mayor, pues se ha multiplicado por cinco, de 1633 en 1991 a 7800 en 2017.

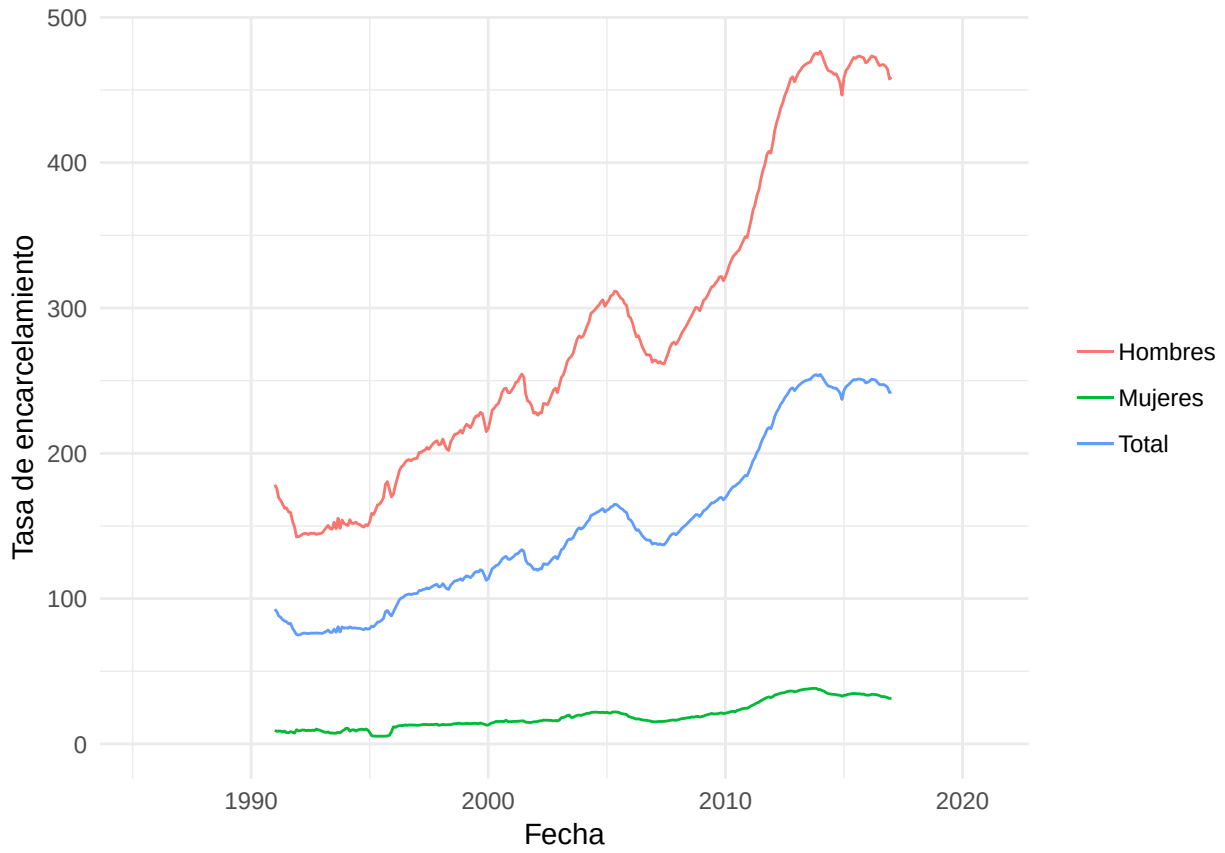



```
## Saving 6.5 x 4.5 in image
```

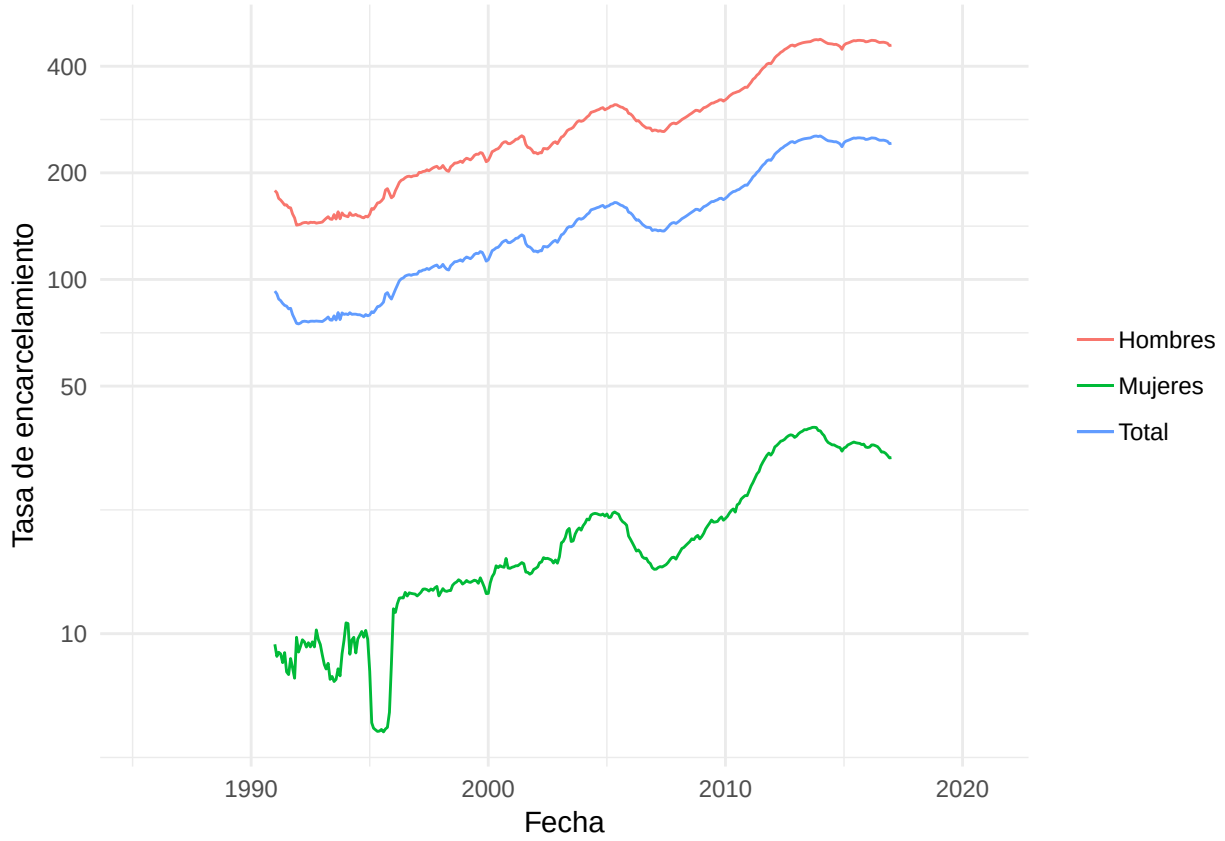



```
## Saving 6.5 x 4.5 in image
```

El incremento en la población carcelaria podría tomarse como un efecto del crecimiento de la población colombiana. Para validar este supuesto calculamos la tasa de encarcelamiento, que mide la cantidad de personas encarceladas por cada cien mil habitantes. Este indicador pasó de 92 personas por cada cien mil habitantes en enero de 1991 a 242 en enero de 2016. Tal incremento se puede ver tanto en hombres como en mujeres.

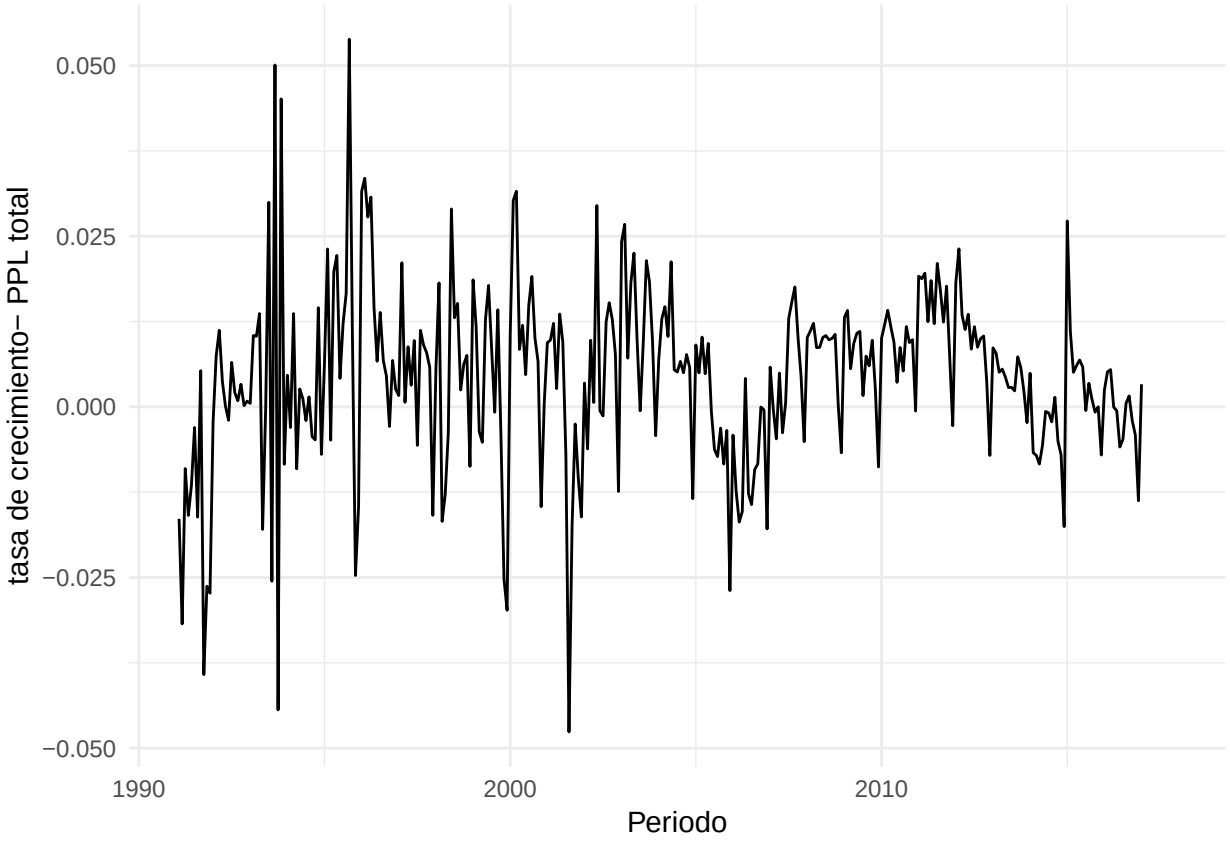


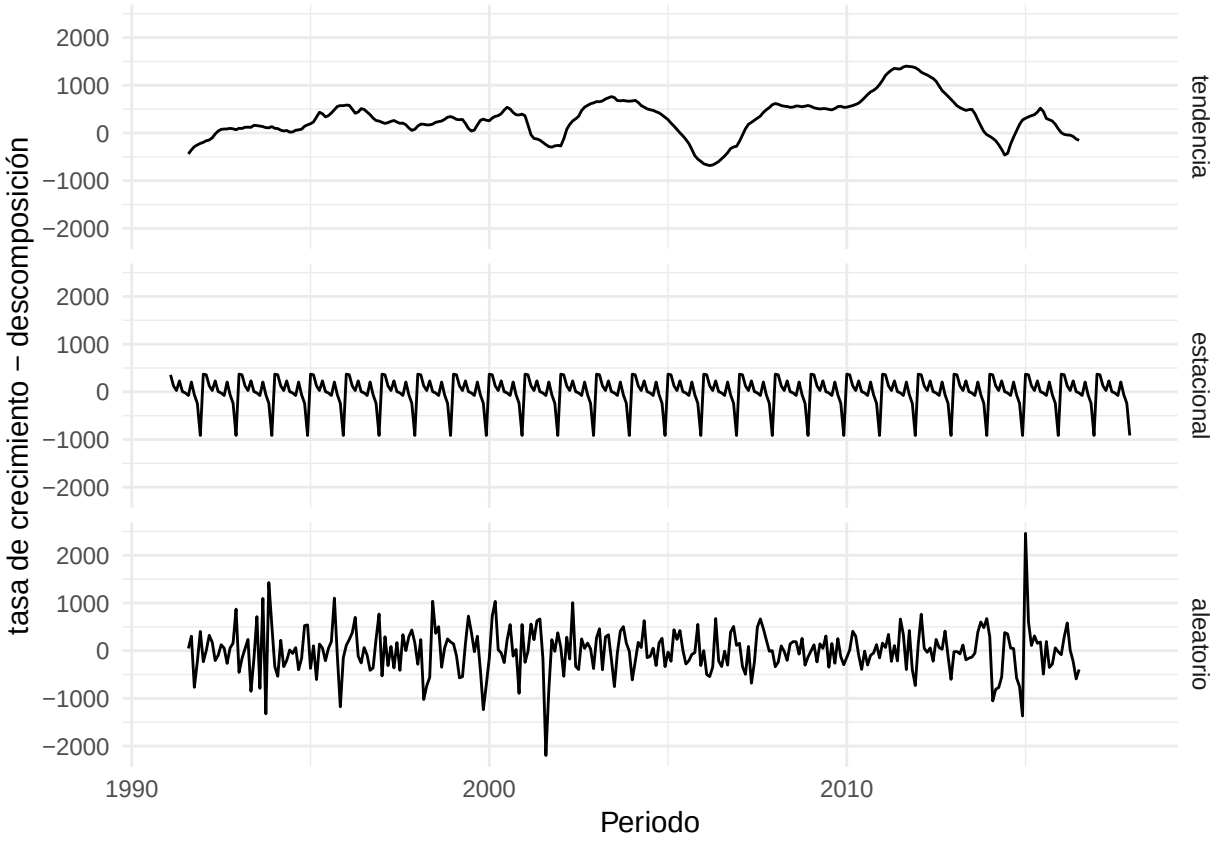
La tasa de encarcelamiento ha crecido de forma exponencial, tanto en hombres como en mujeres, y se puede ver en la gráfica siguiente.

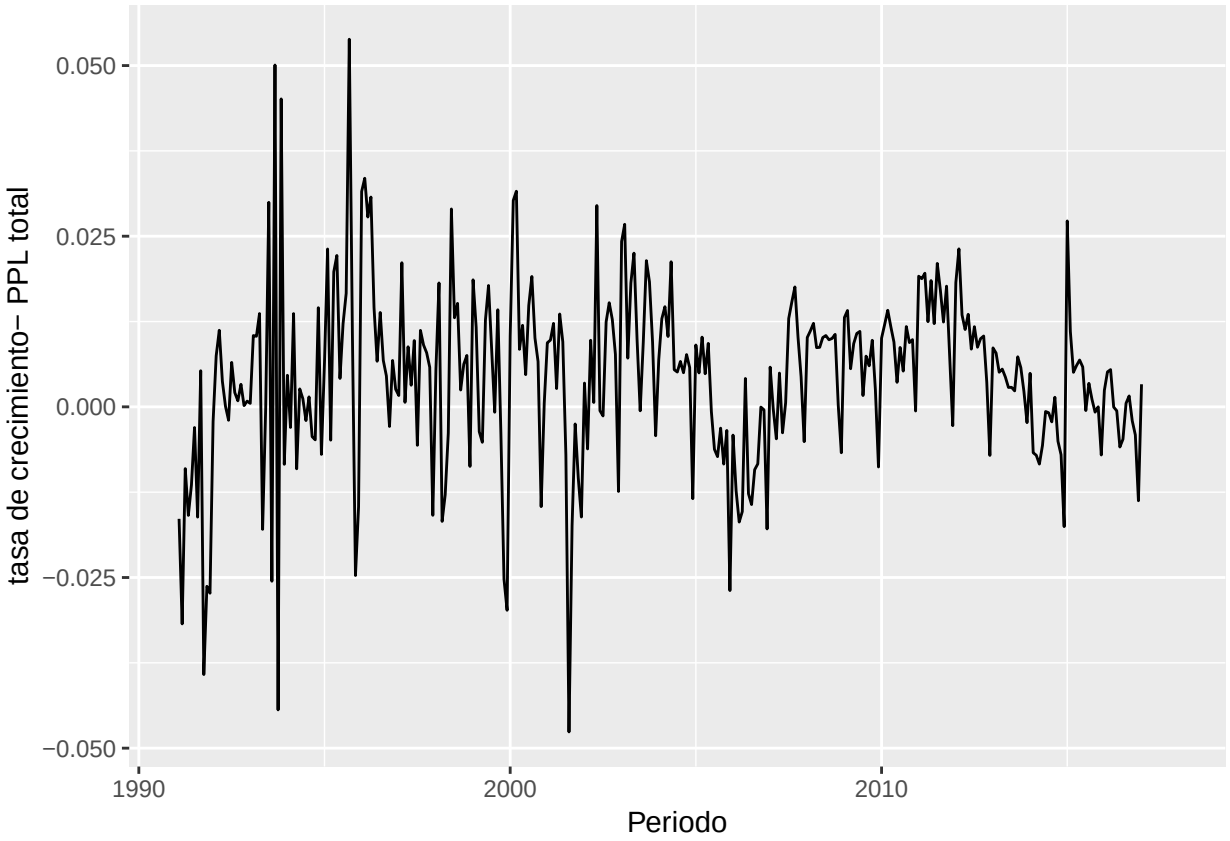


Crecimiento de la Población Privada de la Libertad (PPL)

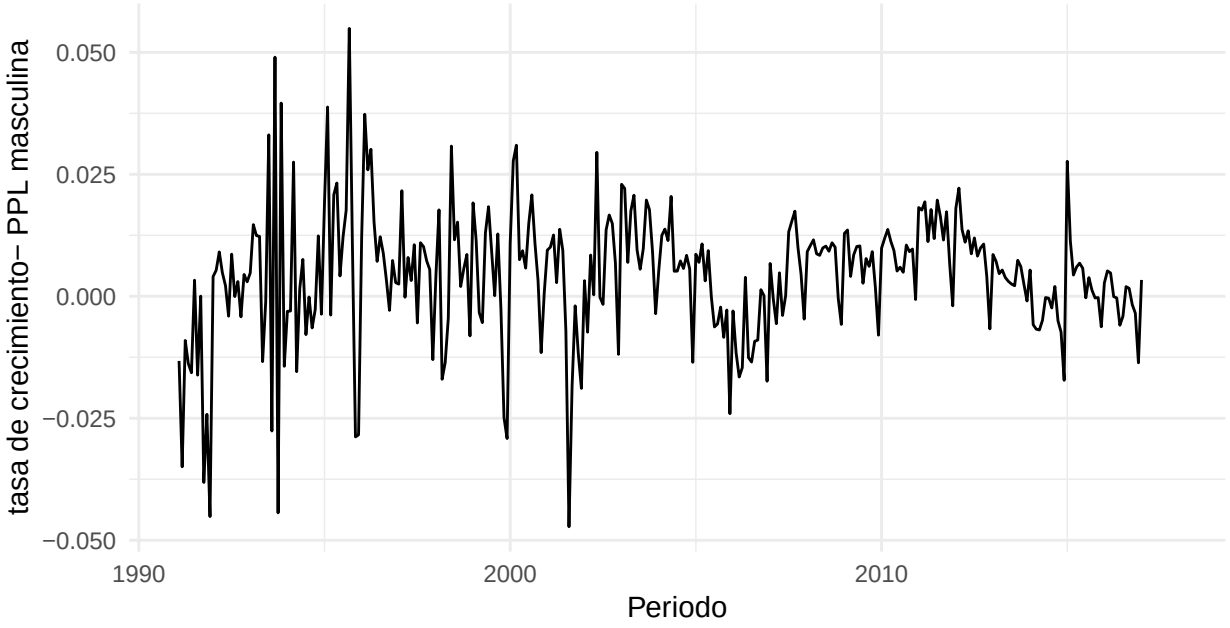
Una primera aproximación al análisis de la población carcelaria, se podría realizar al separar los componentes estacionales, de tendencia y aleatorios de la serie de tiempo. No obstante, es posible inferir que la variabilidad de la serie no es constante. Por esta razón resultaría pertinente analizar la tasa de crecimiento de la población de mes a mes. Una técnica comunmente usada es trabajar con la difencia de los logaritmos de la población, que para variaciones cercanas a cero, se aproxima a la tasa de crecimiento.

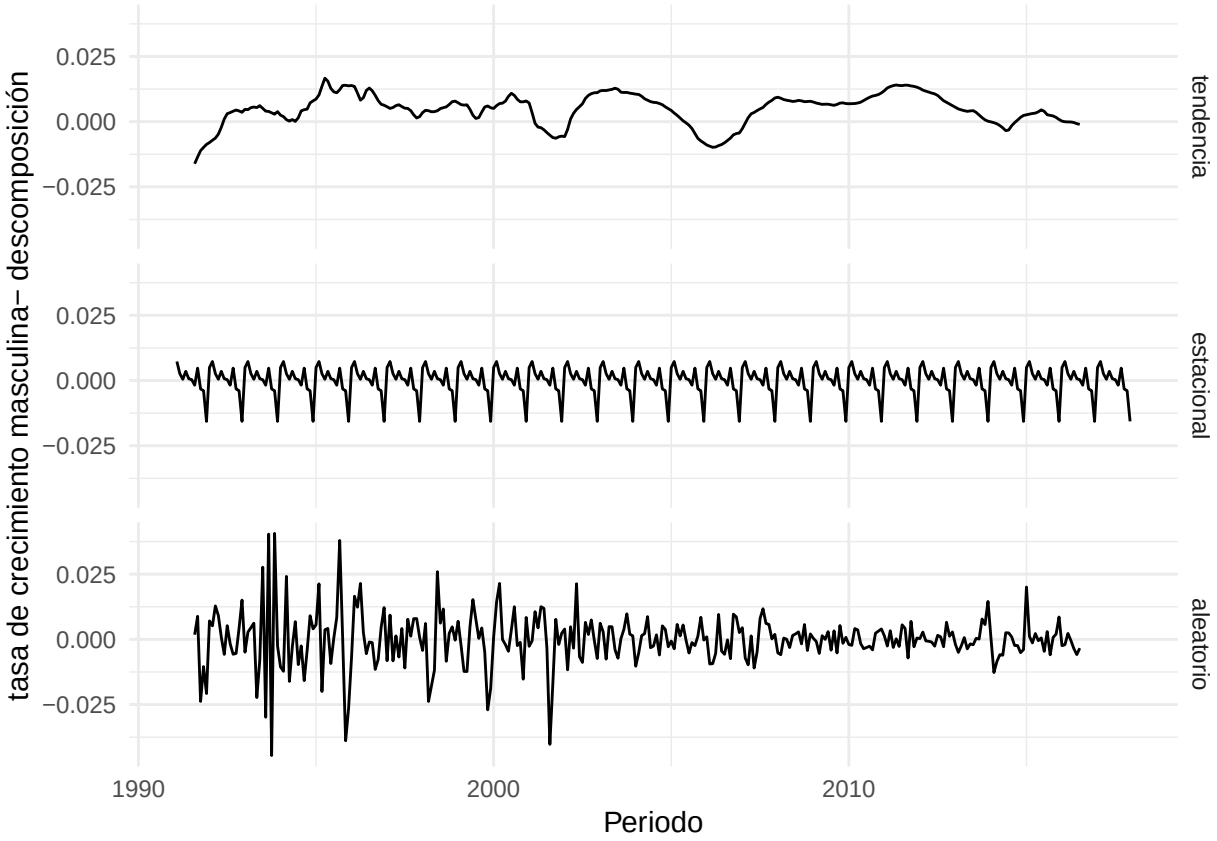

0.050
0.025
0.000
−0.025
−0.050
tasa de crecimiento− PPL total
1990
2000
2010
Periodo


tasa de crecimiento – descomposición
2000
1000
0
−1000
−2000
tendencia
estacional
aleatorio
1990
2000
2010
Periodo

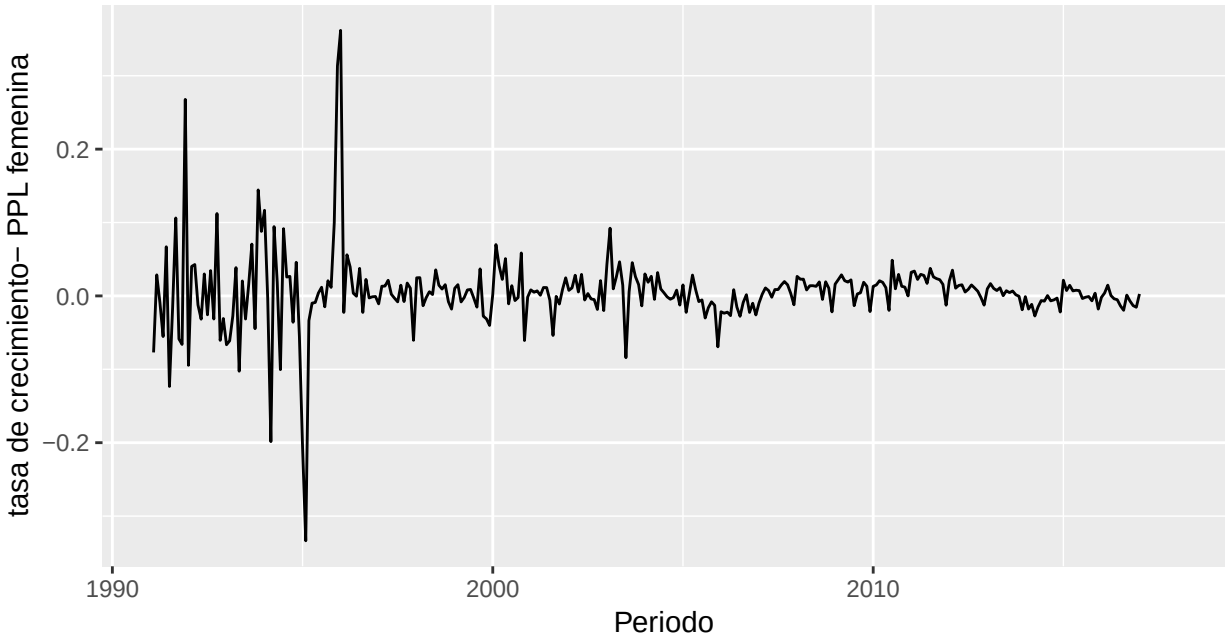


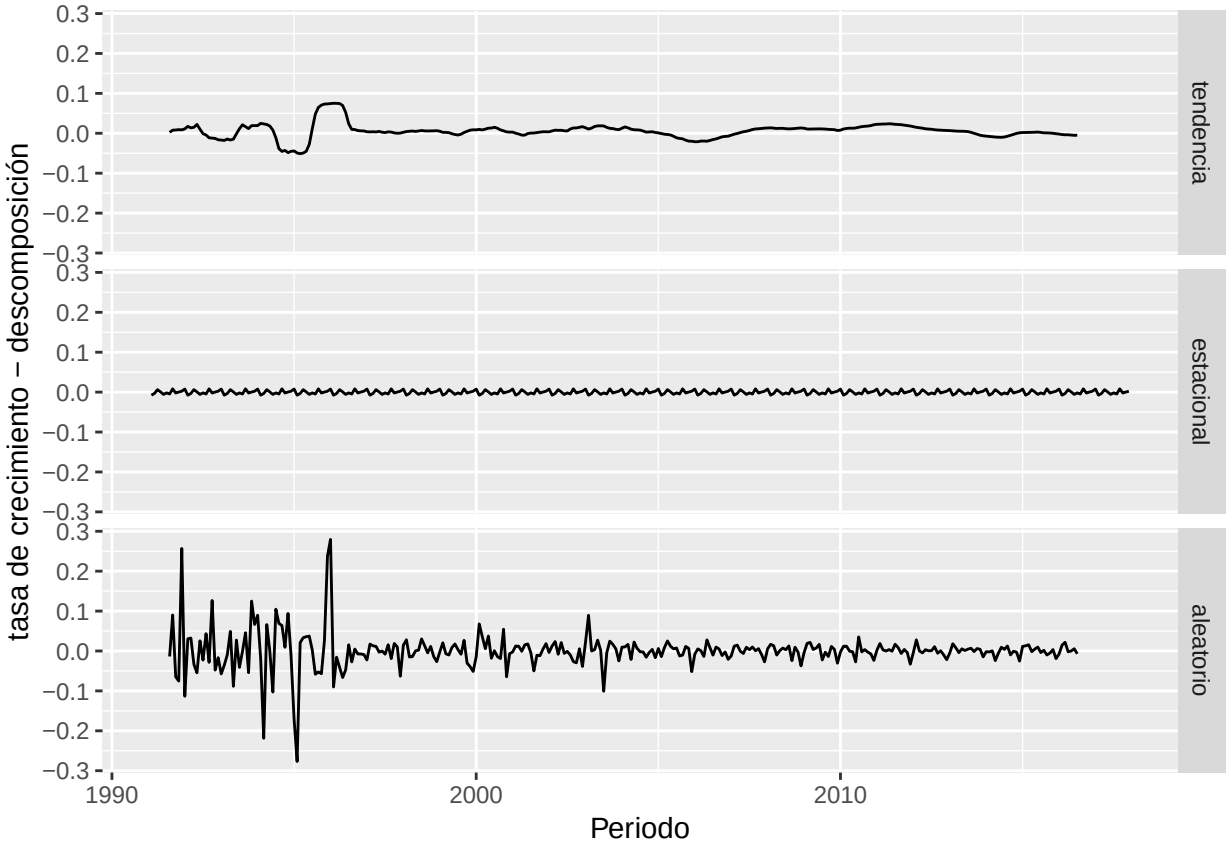
Crecimiento de la población privada de la libertad masculida


tasa de crecimiento− PPL masculina
0.050
0.025
0.000
−0.025
−0.050
1990
2000
2010
Periodo



Crecimiento de la población privada de la libertad femenina


tasa de crecimiento− PPL femenina
0.2
0.0
−0.2
1990
2000
2010
Periodo

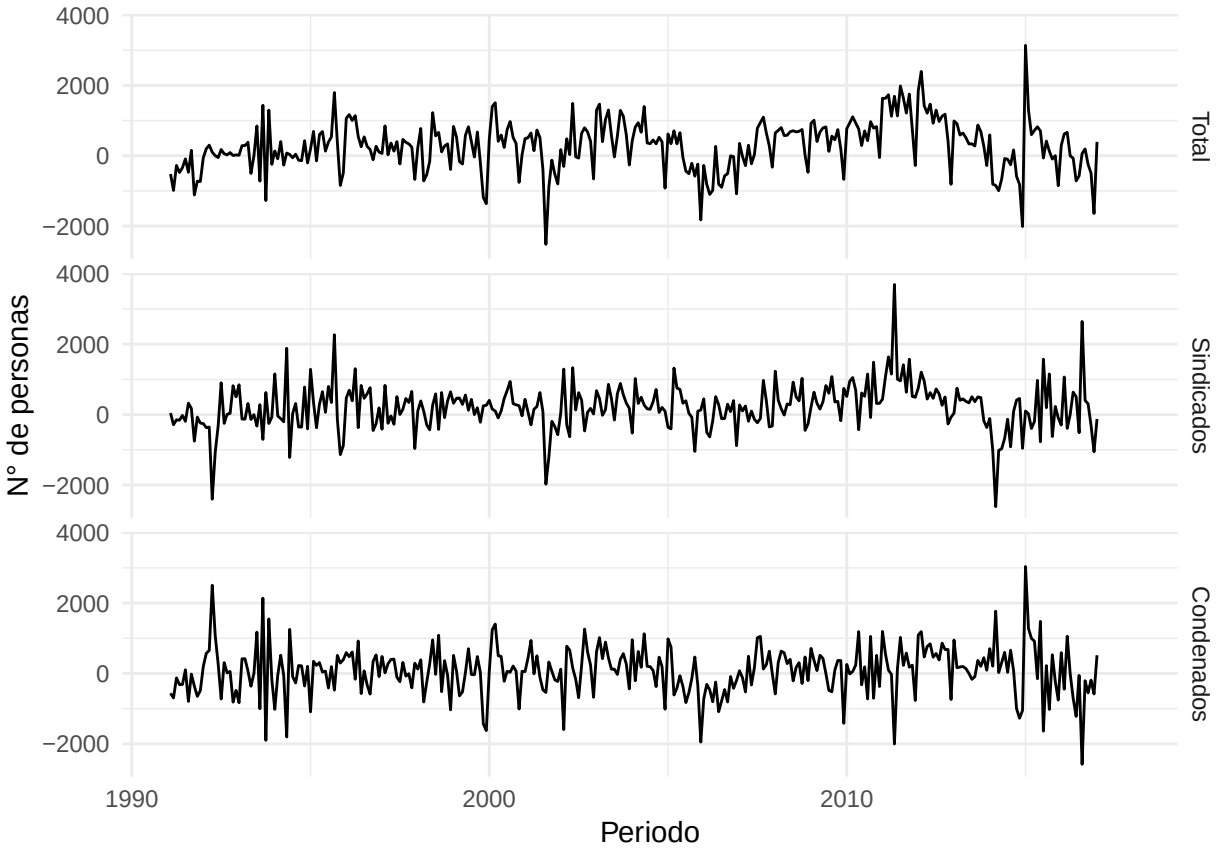


Como se evidencia que componente aleatorio no tiene varianza constante se genera componente lineal, el análisis pertinente es por sindicados y condenados, pues es el que se considera en el resto de la tesis.

```
# Población sindicada a serie de tiempo por situación
ppl_sitjur  %>% spread(key = "categoria", value = valor) -> ppl_situacion
# Serie de tiempo de población total
TPPtimeseries <- ts(ppl_situacion$total, frequency=12, start=c(1991,1))
# Serie de tiempo de población sindicada
SPPtimeseries <- ts(ppl_situacion$condenados, frequency=12, start=c(1991,1))
# Serie de tiempo de población condenada
CPPtimeseries <- ts(ppl_situacion$sindicados, frequency=12, start=c(1991,1))

# Ajustar a zoo, para poder hacer gráfica
Total <- as.zoo(diff(TPPtimeseries))
Sindicados <- as.zoo(diff(SPPtimeseries))
Condenados <- as.zoo(diff(CPPtimeseries))

# Gráficar crecimient de las tres
autoplot(merge(Total, Sindicados, Condenados), geom = "line")  + ylab("N° de personas") + xlab("Periodo
graf_var
```

```
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 11 rows containing missing values (geom_path).
```




```
ggsave("variacion_intermensual.png")
```

```
## Saving 6.5 x 4.5 in image
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 11 rows containing missing values (geom_path).
```

```
# Descomponer variación intermensual
DeltaTPP <- decompose(Total)
```

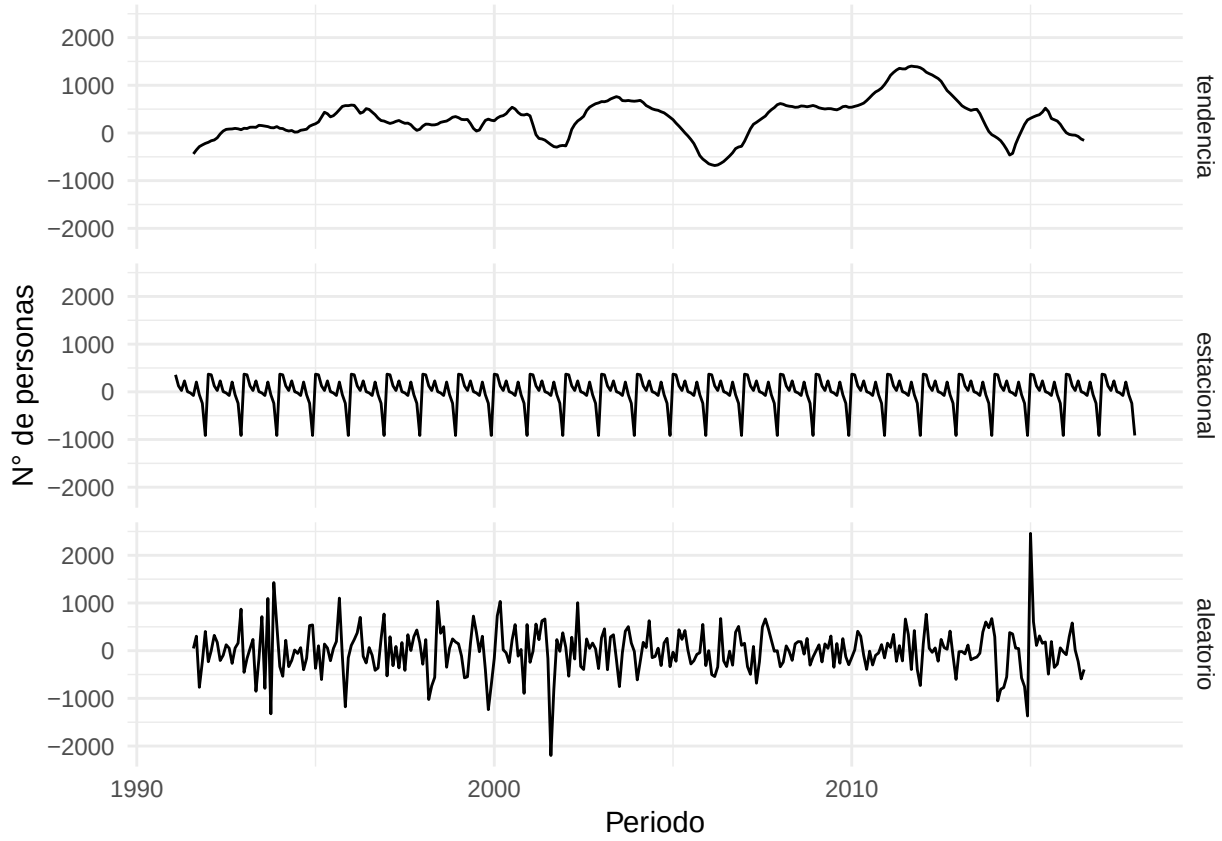
```
## Warning in decompose(Total): Métodos incompatibles ("Ops.zoo", "Ops.ts")
## para "-"
```

```
## Warning in structure(list(x = x, seasonal = seasonal, trend = trend, random
## = if (type == : Métodos incompatibles ("Ops.zoo", "Ops.ts") para "-"
```

```
# Asignar
tendencia <- as.zoo(DeltaTPP$trend)
aleatorio <- as.zoo(DeltaTPP$random)
estacional <- as.zoo(DeltaTPP$seasonal)

# Generar gráfica
autoplot(merge(tendencia,estacional, aleatorio), geom = "line") + ylab("N° de personas") + xlab("Period
graf_tendencia_tpp
```

```
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 23 rows containing missing values (geom_path).
```




```
ggsave("variacion_mensual_total_desc.png")
```

```
## Saving 6.5 x 4.5 in image
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 23 rows containing missing values (geom_path).
```

```
# Descomponer variación intermensual
DeltaSPP <- decompose(Sindicados)
```

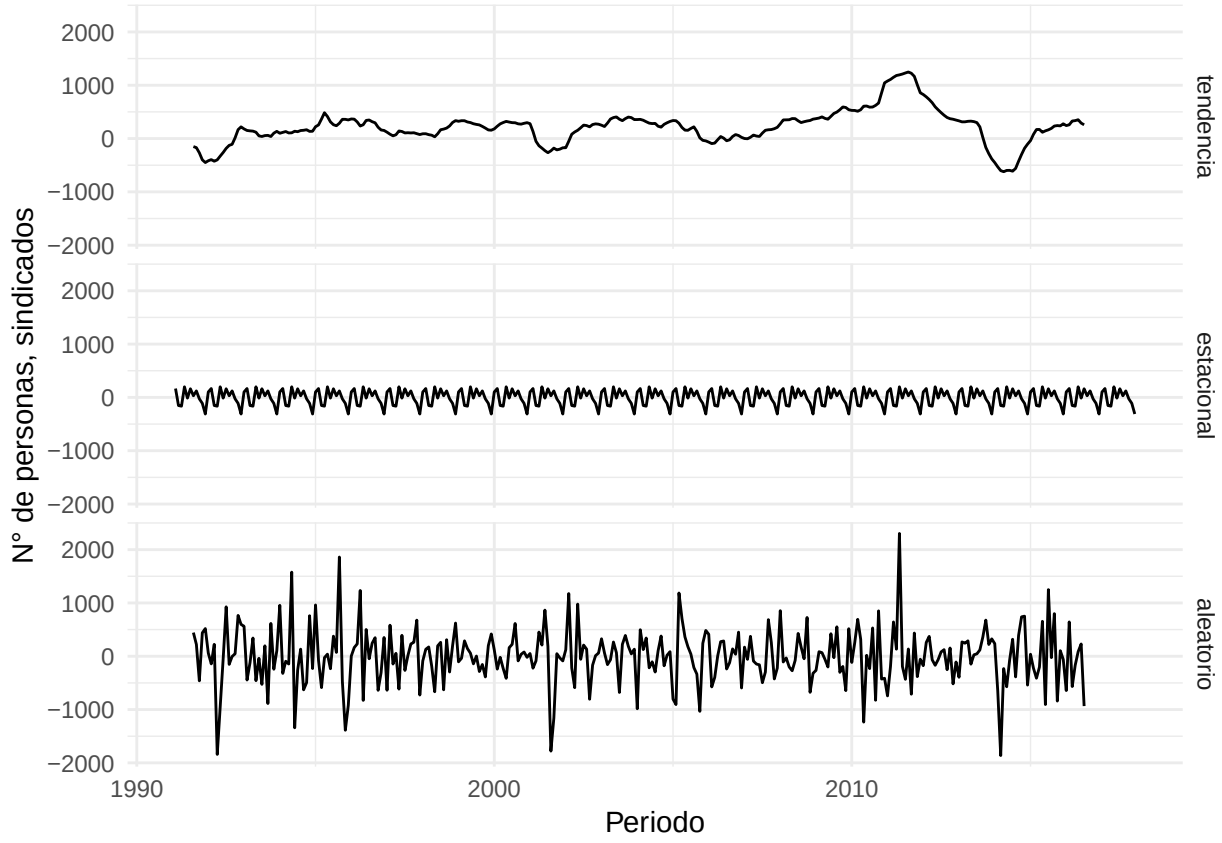
```
## Warning in decompose(Sindicados): Métodos incompatibles ("Ops.zoo",
## "Ops.ts") para "-"
```

```
## Warning in structure(list(x = x, seasonal = seasonal, trend = trend, random
## = if (type == : Métodos incompatibles ("Ops.zoo", "Ops.ts") para "-"
```

```
# Asignar
tendencia <- as.zoo(DeltaSPP$trend)
aleatorio <- as.zoo(DeltaSPP$random)
estacional <- as.zoo(DeltaSPP$seasonal)

# Generar gráfica
autoplot(merge(tendencia,estacional, aleatorio), geom = "line") + ylab("N° de personas, sindicados") + 

# gráfica
graf_tendencia_spp
```

```
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 23 rows containing missing values (geom_path).
```




```
ggsave("variacion_mensual_sindicados_desc.png")
```

```
## Saving 6.5 x 4.5 in image
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
```

```
## Warning: Removed 23 rows containing missing values (geom_path).
```

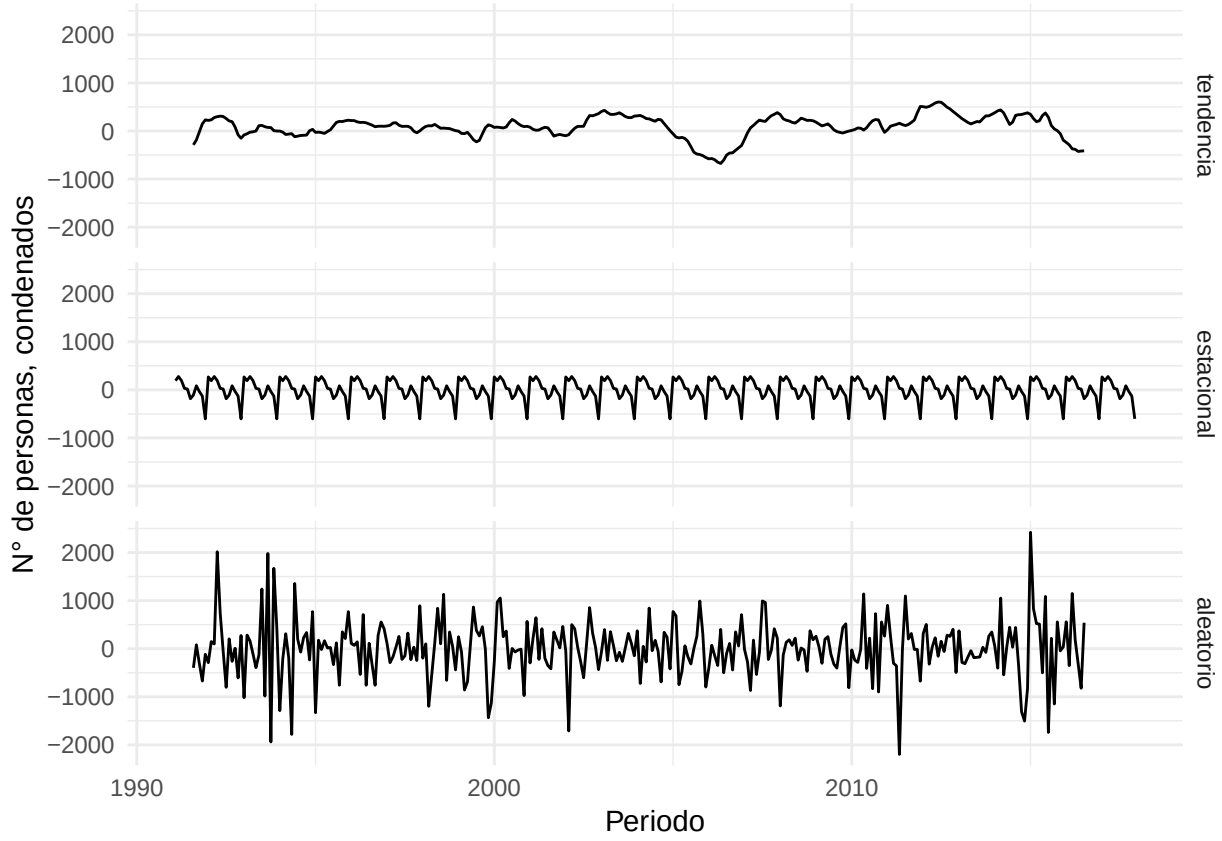
```
# Descomponer variación intermensual
DeltaCPP <- decompose(Condenados)
```

```
## Warning in decompose(Condenados): Métodos incompatibles ("Ops.zoo",
## "Ops.ts") para "-"
```

```
## Warning in structure(list(x = x, seasonal = seasonal, trend = trend, random
## = if (type == : Métodos incompatibles ("Ops.zoo", "Ops.ts") para "-"
```

```
# Asignar
tendencia <- as.zoo(DeltaCPP$trend)
aleatorio <- as.zoo(DeltaCPP$random)
estacional <- as.zoo(DeltaCPP$seasonal)

# Generar gráfica
autoplot(merge(tendencia,estacional, aleatorio), geom = "line") + ylab("N° de personas, condenados") + x

# gráfica
graf_tendencia_cpp
```

```
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
## Warning: Removed 23 rows containing missing values (geom_path).
```




```
ggsave("variacion_mensual_condenados_desc.png")
```

```
## Saving 6.5 x 4.5 in image
## Don't know how to automatically pick scale for object of type yearmon. Defaulting to continuous.
## Warning: Removed 23 rows containing missing values (geom_path).
```

Procesos ARIMA

Los procesos ARMA son procesos aleatorios de la forma

$Y_t = \gamma Y_{t-1} + \gamma_2 Y_{t-2} ... + \epsilon + \theta_1 \epsilon_{t-1}$

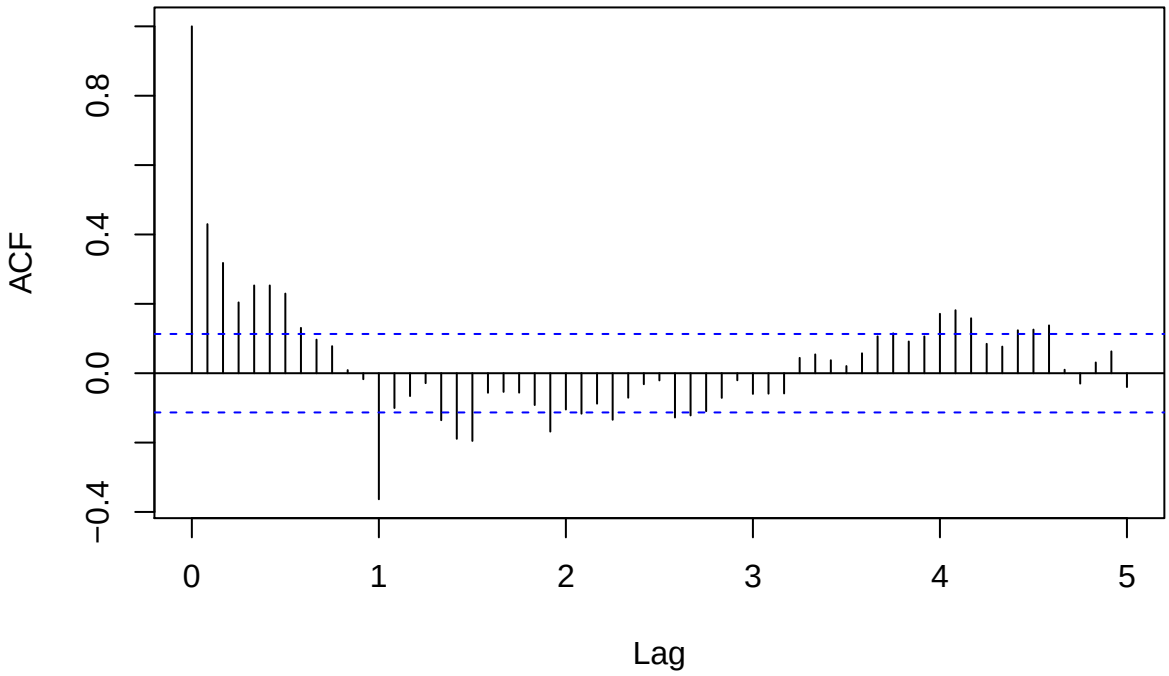
Aunque el termino ϵ no tiene necesariamente una distribución normal, por el resto de documento se asumirá una distribución normal con media μ y varianza σš, a menos que se especifique lo contrario.

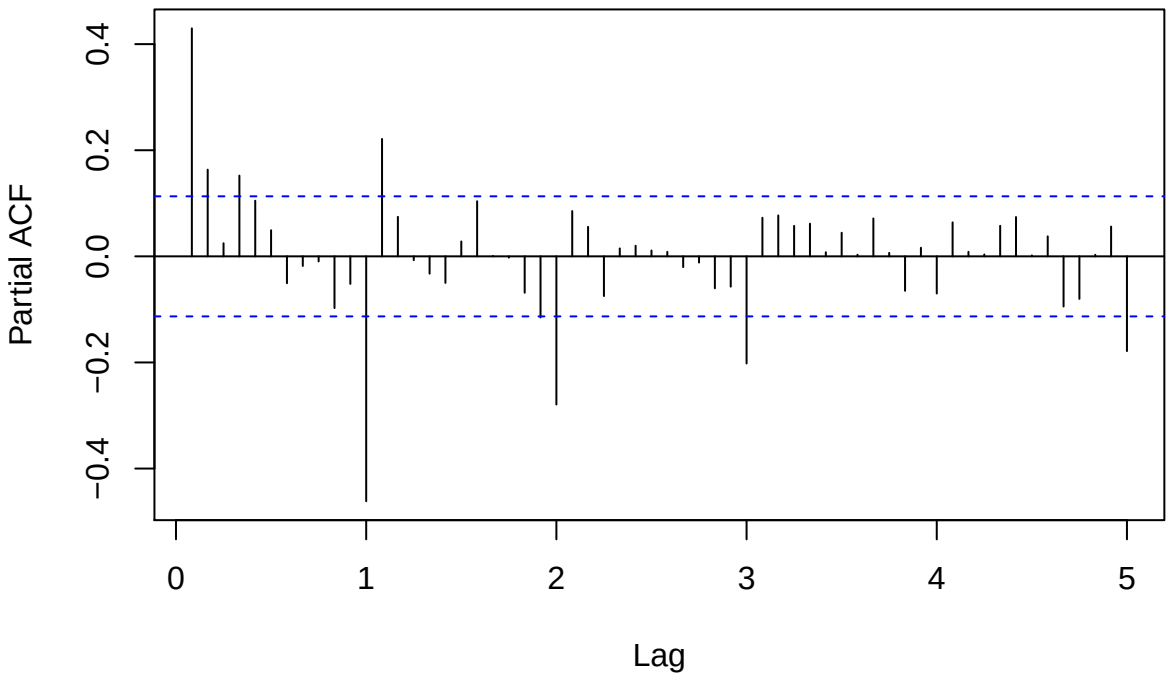
Los procesos ARIMA resultan al considerar una serie de la forma:

$Y_t = \alpha + Y_{t-1}$

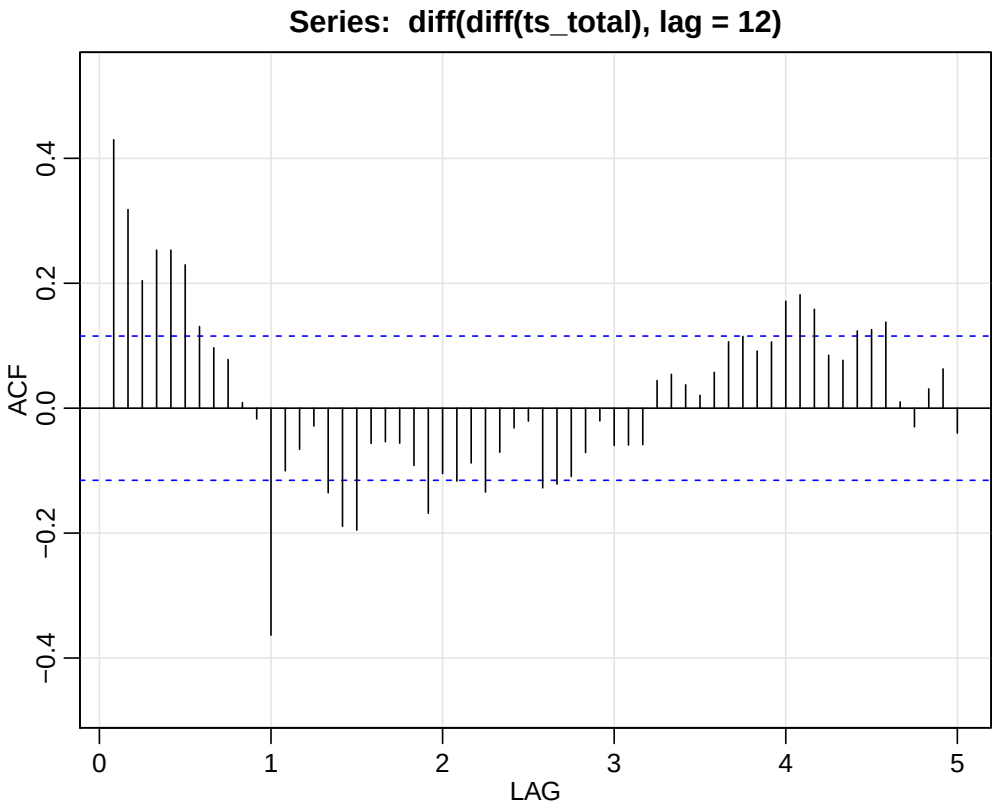
Tal que el proceso $Y_t - Y_{t-1}$ es un proceso ARMA.

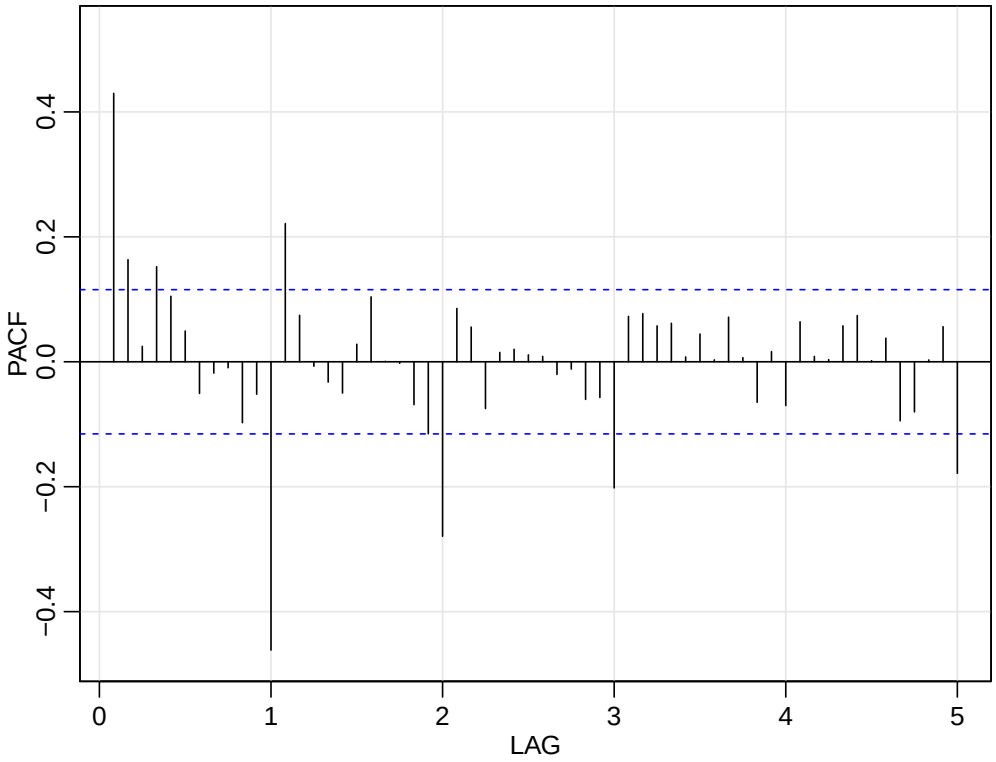
Una primera aproximación a la proyección de poblaciones carcelarias será validar si es posible modelar el proceso como un proceso ARIMA. Con este propósito presentamos las funciones de autocorrelación y autocorrelación parcial de la población total.


ACF
0.8
0.4
0.0
−0.4
0
1
2
3
4
5
Lag

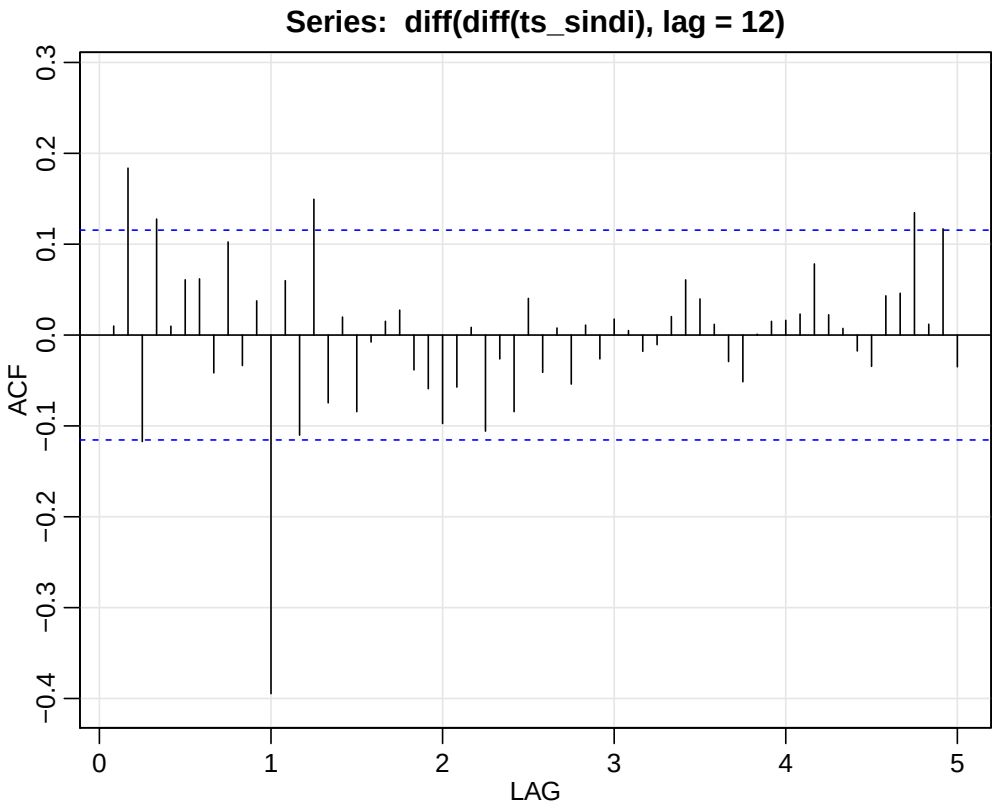


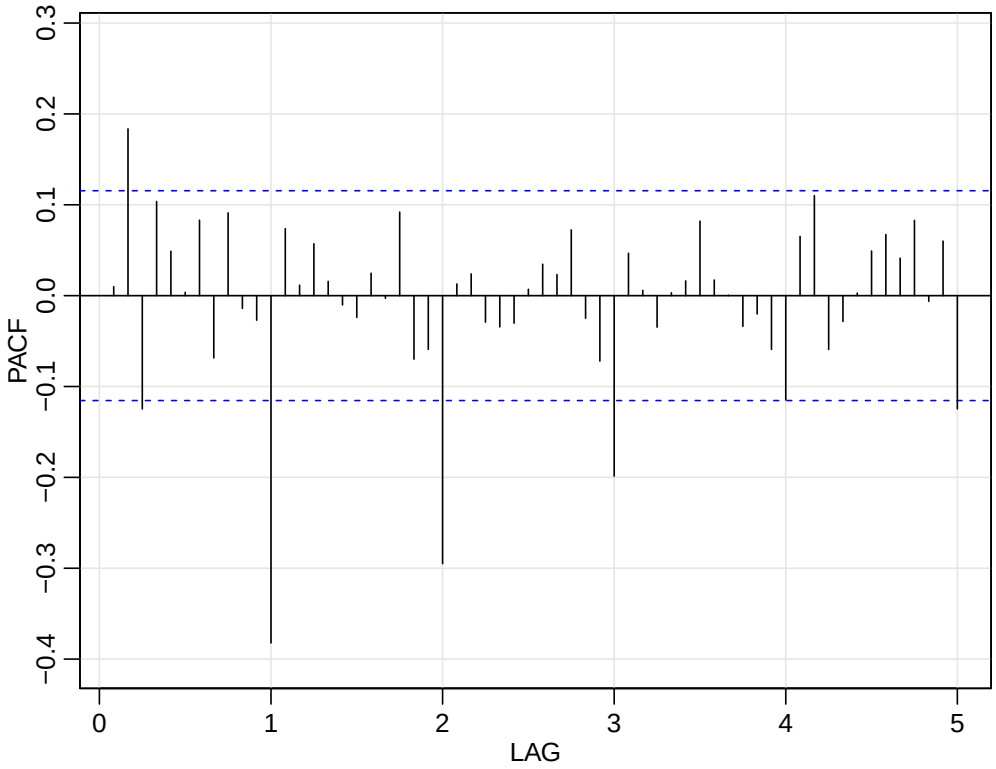

```
## Crear variables
# Población carcelaria total
ppl_sitjur %>% dplyr::filter(categoria == "total", !(is.na(valor))) -> ppl_total
ppl_total <- ppl_total$valor
ts_total <- ts(ppl_total, start = 1991, frequency = 12)
# Sindicados
ppl_sitjur %>% dplyr::filter(categoria == "sindicados", !(is.na(valor))) -> ppl_sindi
ppl_sindi <- ppl_sindi$valor
ts_sindi <- ts(ppl_sindi, start = 1991, frequency = 12)
# Condenados
ppl_sitjur %>% dplyr::filter(categoria == "condenados", !(is.na(valor))) -> ppl_conde
ppl_conde <- ppl_conde$valor
ts_conde <- ts(ppl_conde, start = 1991, frequency = 12)
error = 2/sqrt(length(ppl_conde))

## Población total
# Calcular acf y pacf
acf_total <- as.data.frame(acf2(diff(diff(ts_total), lag = 12), max.lag = 60))
```


Series: diff(diff(ts_total), lag = 12)
ACF
0.4
0.2
0.0
−0.2
−0.4
0
1
2
3
4
5
LAG

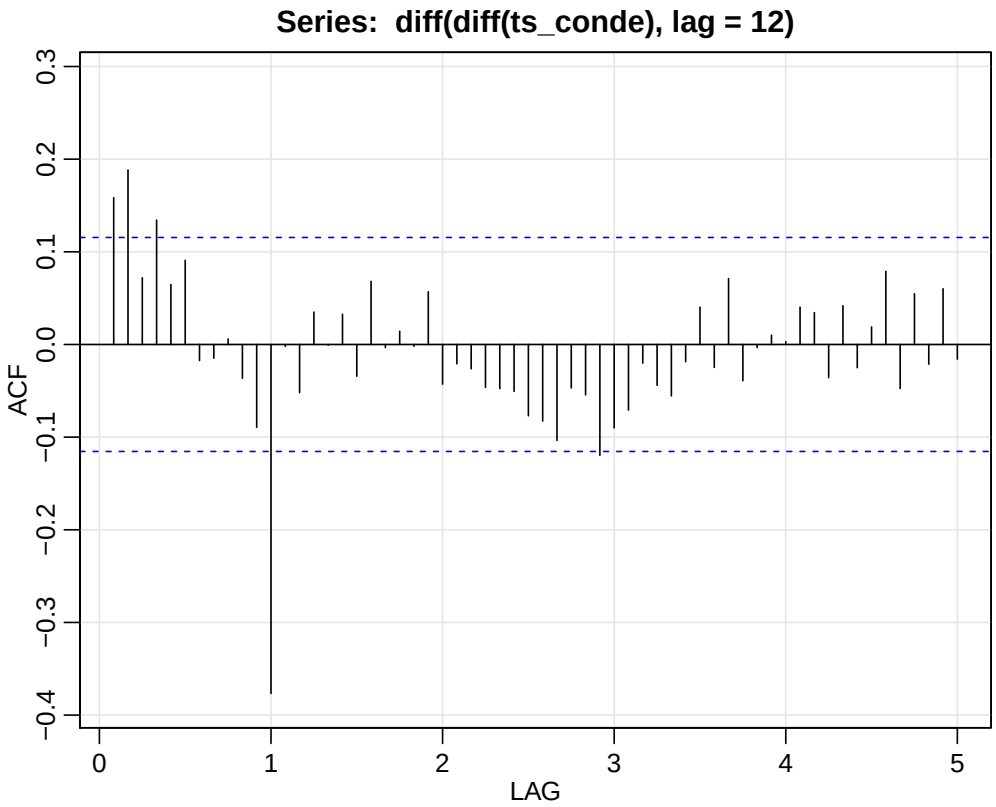

PACF
0.4
0.2
0.0
−0.2
−0.4
0
1
2
3
4
5
LAG

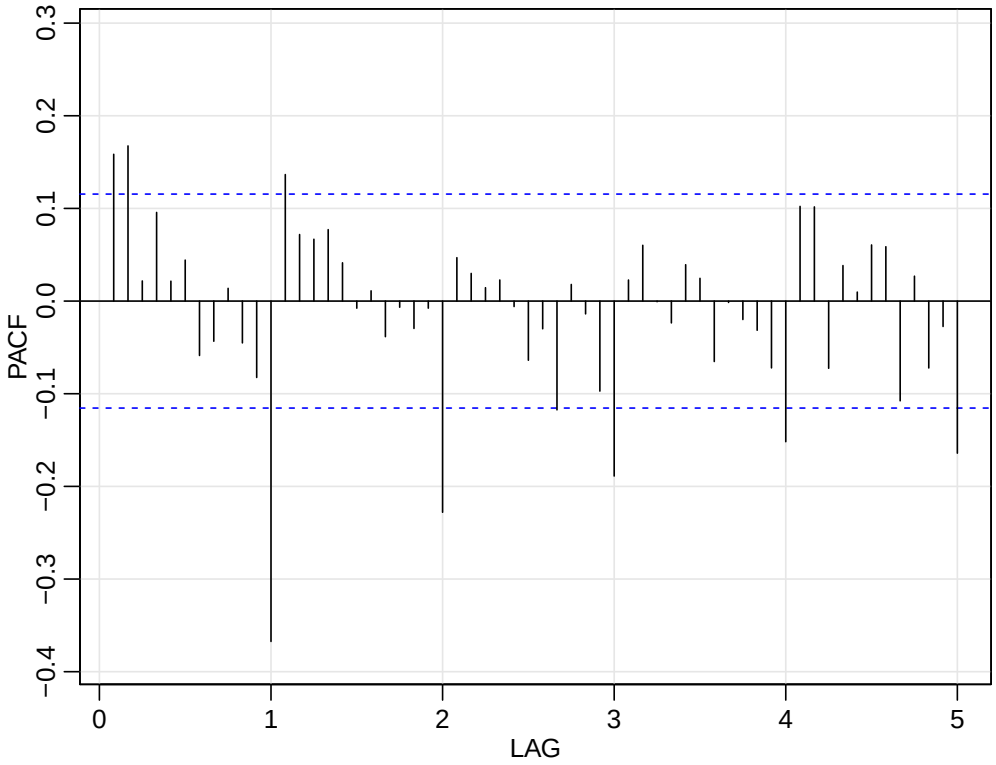
```r
# Gráfica acf total
acf_total %>% ggplot() + geom_bar(aes(x = c(1:60), y = ACF), stat = "identity", fill = "steelblue") + s
# Gráfica pacf total
acf_total %>% ggplot() + geom_bar(aes(x = c(1:60), y = PACF), stat = "identity", fill = "steelblue") + s

## Población sindicada
# Calcular acf y pacf
acf_sindi <- as.data.frame(acf2(diff(diff(ts_sindi), lag = 12), max.lag = 60))
```


Series: diff(diff(ts_sindi), lag = 12)
ACF
LAG
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
1
2
3
4
5

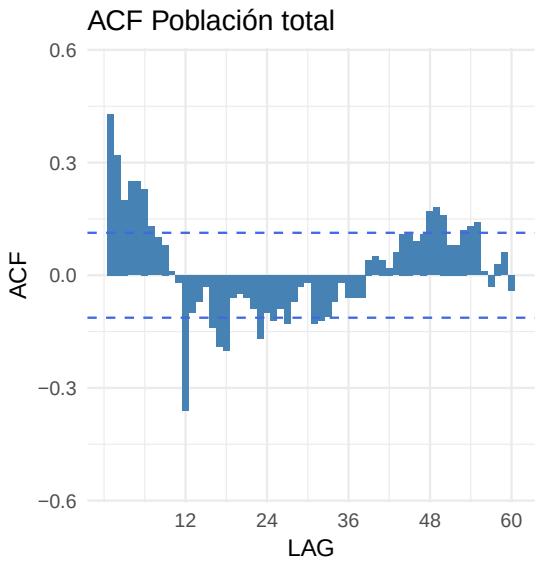

PACF
LAG
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
1
2
3
4
5

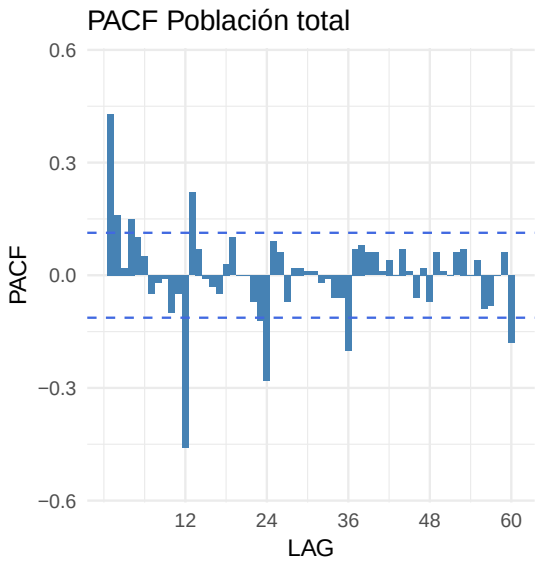
```r
# Gráfica acf sindicados
acf_sindi %>% ggplot() + geom_bar(aes(x = c(1:60), y = ACF), stat = "identity", fill = "steelblue1") + s
# Gráfica pacf sindicados
acf_sindi %>% ggplot() + geom_bar(aes(x = c(1:60), y = PACF), stat = "identity", fill = "steelblue1") +

## Población condenada
# Calcular acf y pacf
acf_conde <- as.data.frame(acf2(diff(diff(ts_conde), lag = 12), max.lag = 60))
```

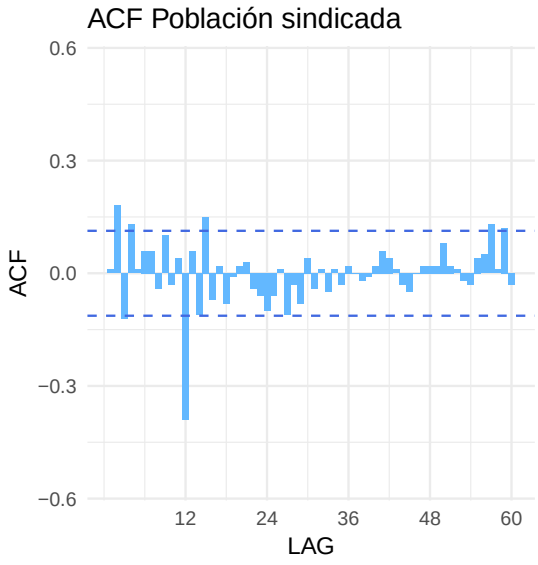

Series: diff(diff(ts_conde), lag = 12)
ACF
LAG
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
1
2
3
4
5

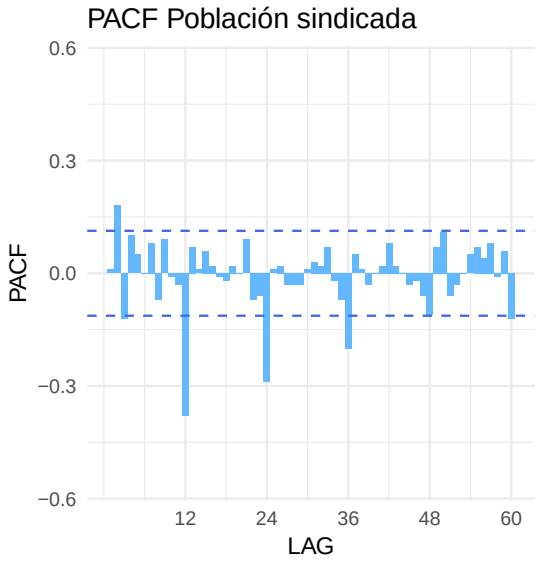

PACF
LAG
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
1
2
3
4
5

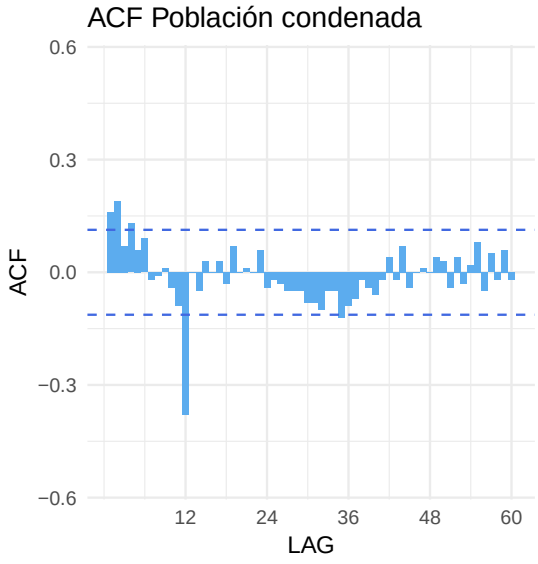
```r
# Gráfica acf condenados
acf_conde %>% ggplot() + geom_bar(aes(x = c(1:60), y = ACF), stat = "identity", fill = "steelblue2") + s

# Gráfica pacf condenados
acf_conde %>% ggplot() + geom_bar(aes(x = c(1:60), y = PACF), stat = "identity", fill = "steelblue2") +

#png(file = 'ACF_var_pob.png', height = 750, width = 500, res = 85)
ACF_var_pob <- grid.arrange(graf_acf_total, graf_pacf_total,graf_acf_sindi, graf_pacf_sindi,graf_acf_co
```

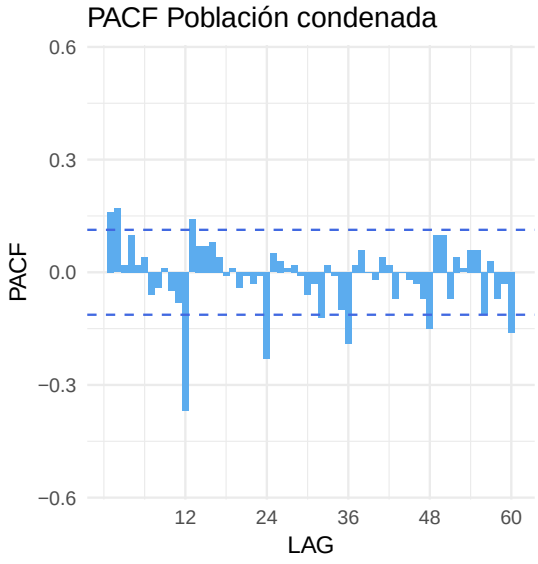

ACF Población total
ACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60


PACF Población total
PACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60


ACF Población sindicada
ACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60


PACF Población sindicada
PACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60


ACF Población condenada
ACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60


PACF Población condenada
PACF
LAG
0.6
0.3
0.0
−0.3
−0.6
12
24
36
48
60

```
ACF_var_pob
```

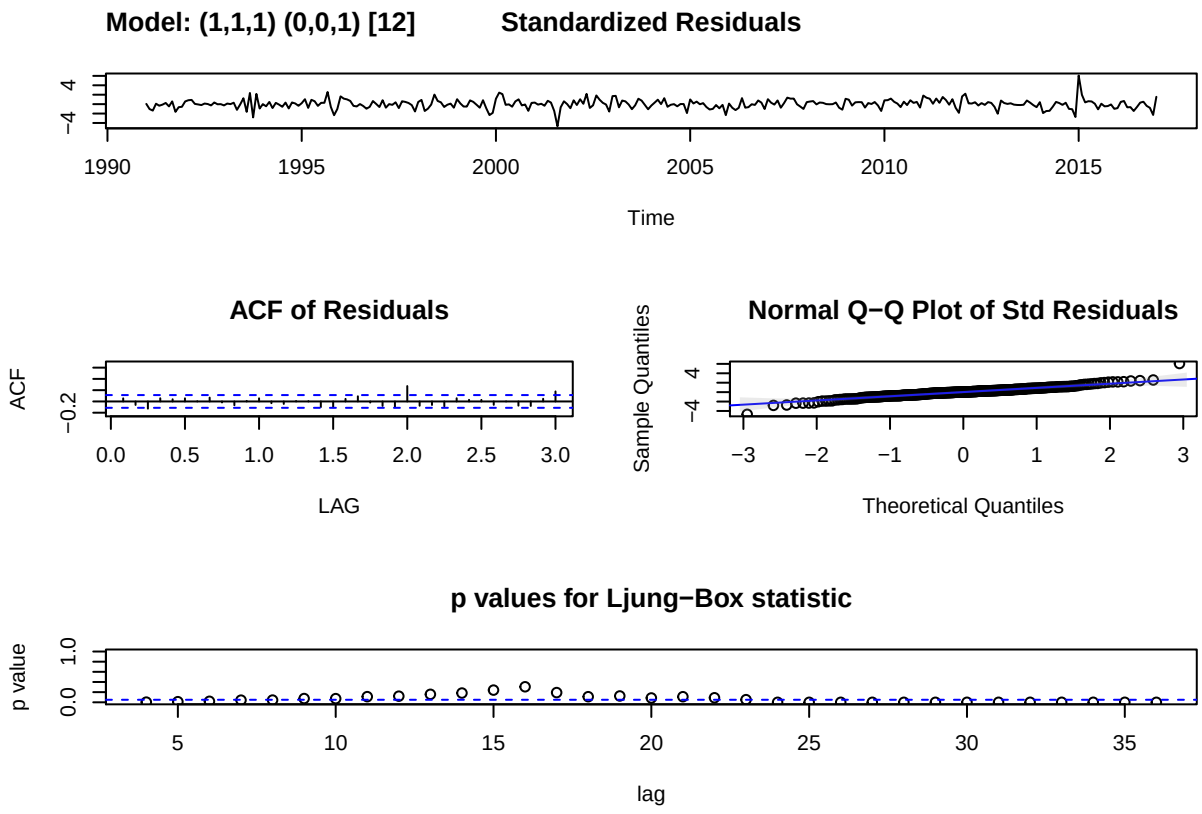
```
## TableGrob (3 x 2) "arrange": 6 grobs
##   z     cells    name           grob
## 1 1 (1-1,1-1) arrange gtable[layout]
## 2 2 (1-1,2-2) arrange gtable[layout]
## 3 3 (2-2,1-1) arrange gtable[layout]
## 4 4 (2-2,2-2) arrange gtable[layout]
## 5 5 (3-3,1-1) arrange gtable[layout]
## 6 6 (3-3,2-2) arrange gtable[layout]
```

```
#dev.copy(png,'ACF_var_pob')
#dev.off()
```

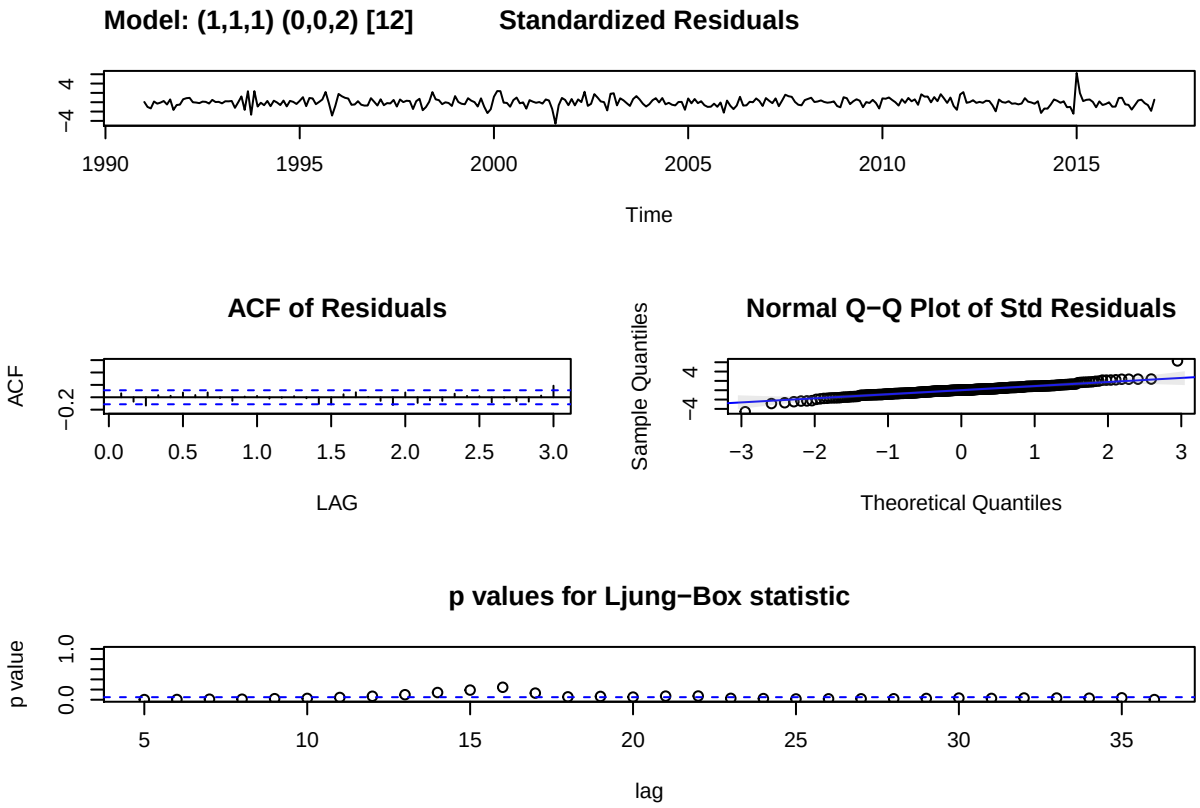
Las funciones de autocorrelación y correlación cruzada, de $Y_t - Y_{t - 1} $ muestran:

- La función de autocorrelación decae, y la función de autocorrelación parcial también decae, sugiriendo un ARMA.
- La función de autocorrelación se encuetra sobre las dos desviaciones estándar, sugiriendo que son significativas.
- La función de autocorrelación tiene un único pico en el periodo doce, la función de autocorrelación parcial decae lentamente, sugiriendo un MA(1)

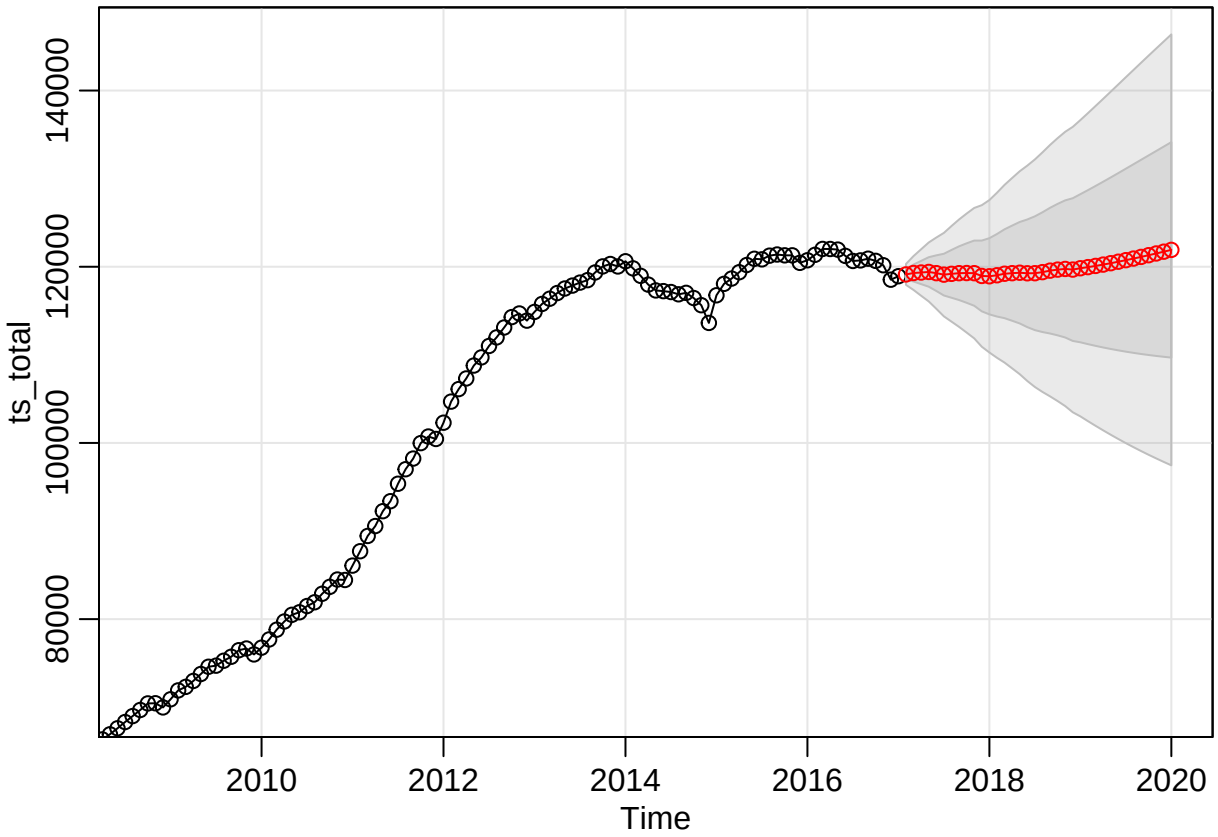
En estas condiciones podemos ajustar un ARIMA (1,1,1,0,0,1)



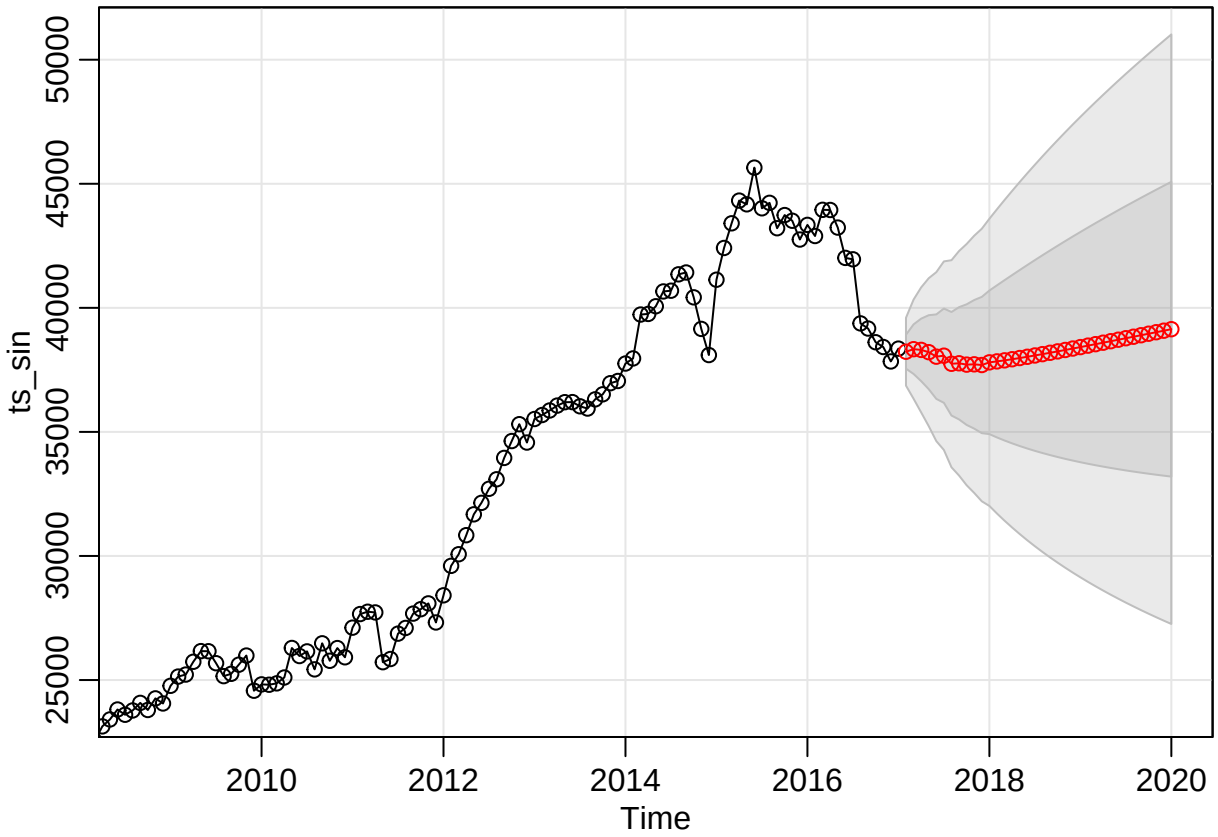
Todos los parámetros del modelo tienen un p_value que sugiere que son significativos, excepto por la constante. Sin embargo, la autocorrelación del periodo dos sugiere que podemos estimar un ARIMA (1,1,1,0,0,2)



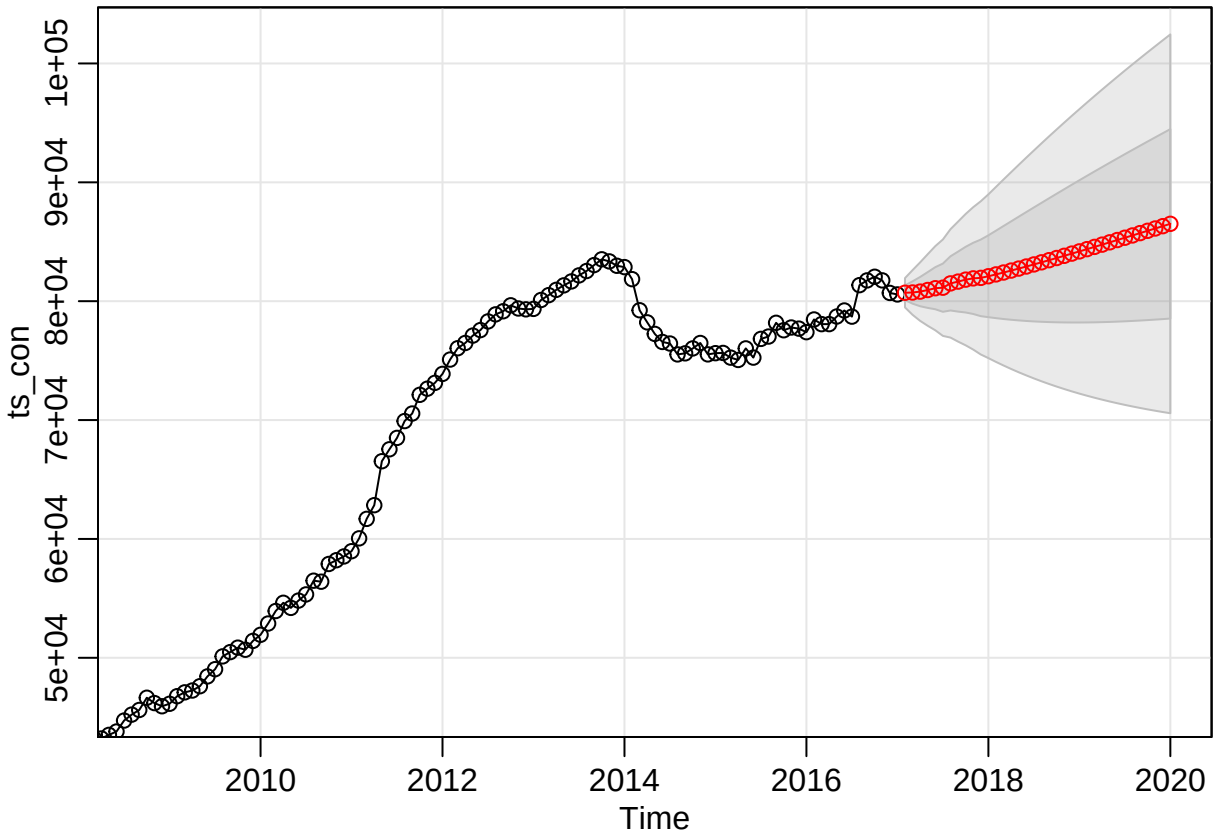
El nuevo término SA2 resulta siginicativo y presenta criterios de información de valor más pequeño, lo que implica que el modelo ARIMA (1,1,1,0,0,2) resulta más apropiado que el modelo ARIMA (1,1,1,0,0,1). Proyecciones de la población carcelaria del modelo ARIMA (1,1,1,0,0,2) se presentan a continuación:



Se presenta proyección de la población sindicada:



Se presenta proyección de la población condenada:



Limitaciones de los modelos ARIMA en proyección de poblaciones carcelarias

Aunque es posible seguir analizando modelos ARIMA hasta encontrar el que presente mejores criterios de información, detendremos el analisis en este punto, para centrarnos en las limitaciones del enfoque.

1. La serie de población carcelarias incluye, por lo menos, dos series de tiempo con un comportamiento diferente: La población condenada y la población sindicada. Estos procesos, son en principio procesos autoregresivos, donde además se podría intuir el error no es independiente (como en la población total).
2. Los shocks en el sistema no son necesariamente estables en el tiempo. Las variaciones en los parámetros del modelo dependen de variables exógenas, como la cantidad de personas que ingresan al sistema y la dureza de las penas.
3. Los intervalos de confianza crecen rápidamente pues el modelo se ve afectado fuertemente por cambios en el nivel.
4. Puesto que el modelo supone unos parámetros fijos no da cuenta de la influencia de variables exógenas en el crecimiento de la población carcelaria.

```
#####Modelos VAR
# ts_sincon <- cbind (diff(ts_sin), diff(ts_con))
# VARselect(ts_sincon, lag.max=9, type="const")
# ts_sincon_mod <- VAR(ts_sincon,p = 1)
# forecast_var <- predict(ts_sincon_mod, n.ahead = 60)
# ts_sincon_mod
# forecast_var
#
# # Cite packages
# if(nchar(system.file(package="astsa"))) citation("astsa")
```

Documento tesis

Estimaciones

Población total

```
### Total 111002
png(file = 'Arima_total_111002.png', height = 750, width = 500, res = 85)
arima_total_1 <- sarima(ts_total,1,1,1,0,0,2, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
### Total 111001
png(file = 'Arima_total_111001.png', height = 750, width = 500, res = 85)
arima_total_2 <- sarima(ts_total,1,1,1,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
auto.arima(ts_total)
```

```
## Series: ts_total
## ARIMA(3,2,1)(0,0,2)[12]
##
## Coefficients:
##          ar1      ar2      ar3      ma1    sma1    sma2
##       0.0520  -0.0864  -0.1553  -0.6756  0.2357  0.1908
## s.e.  0.1034   0.0748   0.0712   0.0931  0.0575  0.0491
##
## sigma^2 estimated as 363346:  log likelihood=-2430.23
## AIC=4874.46   AICc=4874.83   BIC=4900.64
```

```
### Total Auto Arima
png(file = 'Arima_total_321002.png', height = 750, width = 500, res = 85)
arima_total_3 <- sarima(ts_total,3,2,1,0,0,2, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
auto.arima(ts_total, ic = "aic")
```

```
## Series: ts_total
## ARIMA(3,2,1)(0,0,2)[12]
##
## Coefficients:
##          ar1      ar2      ar3      ma1    sma1    sma2
##       0.0520  -0.0864  -0.1553  -0.6756  0.2357  0.1908
## s.e.  0.1034   0.0748   0.0712   0.0931  0.0575  0.0491
##
## sigma^2 estimated as 363346:  log likelihood=-2430.23
## AIC=4874.46   AICc=4874.83   BIC=4900.64
```

```
### Total 121002
png(file = 'Arima_total_121002.png', height = 750, width = 500, res = 85)
arima_total_4 <- sarima(ts_total,1,2,1,0,0,2, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
Indicadores <- matrix(c(arima_total_1$AIC,arima_total_1$BIC,arima_total_2$AIC,arima_total_2$BIC,arima_t

AIC_total <- AIC(arima_total_1$fit,arima_total_2$fit, arima_total_3$fit,arima_total_4$fit)
```

```
## Warning in AIC.default(arima_total_1$fit, arima_total_2$fit,
## arima_total_3$fit, : models are not all fitted to the same number of
## observations
```

```
BIC_total <- BIC(arima_total_1$fit,arima_total_2$fit, arima_total_3$fit,arima_total_4$fit)
```

```
## Warning in BIC.default(arima_total_1$fit, arima_total_2$fit,
## arima_total_3$fit, : models are not all fitted to the same number of
## observations
```

```
Indicadores_df <- cbind(c("(1,1,1,0,0,2)","(1,1,1,0,0,1)","(3,2,1,0,0,2)","(1,2,1,0,0,2)"),round(AIC_to

colnames(Indicadores_df) <- c("ORDEN","AIC","BIC")
#xtable(Indicadores_df)
#xtable(arima_total_4$ttable)
#xtable(arima_total_4$ttable, label = "parámetros_121002", caption = "Parámetros del modelo (1,2,1,0,0,
```

Población sindicada

```
### sindicado 111001
png(file = 'Arima_sindi_11101.png', height = 750, width = 500, res = 85)
arima_sindi_1 <- sarima(ts_sindi,1,1,1,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
### sindicado 211001
png(file = 'Arima_sindi_211001.png', height = 750, width = 500, res = 85)
arima_sindi_2 <- sarima(ts_sindi,2,1,1,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
auto.arima(ts_sindi, ic = "aic")
```

```
## Series: ts_sindi
## ARIMA(0,1,2)(0,0,1)[12]
##
## Coefficients:
##          ma1    ma2    sma1
##       0.0611  0.160  0.1547
## s.e.  0.0572  0.053  0.0571
##
```

```
## sigma^2 estimated as 467153:  log likelihood=-2477.86
## AIC=4963.72   AICc=4963.85   BIC=4978.7
```

```
### sindicado Auto Arima
png(file = 'Arima_sindi_012001.png', height = 750, width = 500, res = 85)
arima_sindi_3 <- sarima(ts_sindi,0,1,2,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
### Total 121002
png(file = 'Arima_sindi_011000.png', height = 750, width = 500, res = 85)
arima_sindi_4 <- sarima(ts_sindi,0,1,1,0,0,0, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
AIC_sindi <- AIC(arima_sindi_1$fit,arima_sindi_2$fit, arima_sindi_3$fit,arima_sindi_4$fit)
BIC_sindi <- BIC(arima_sindi_1$fit,arima_sindi_2$fit, arima_sindi_3$fit,arima_sindi_4$fit)

Indicadores_df_sindi <- cbind(c("(1,1,1,0,0,1)","(2,1,1,0,0,1)","(0,1,2,0,0,1)","(0,1,1,0,0,0)"),round(A

colnames(Indicadores_df_sindi) <- c("ORDEN","AIC","BIC")
xtable(Indicadores_df_sindi)
```

```
## % latex table generated in R 3.5.1 by xtable 1.8-3 package
## % Sun Nov 11 14:00:55 2018
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlll}
##   \hline
##  & ORDEN & AIC & BIC \\
##   \hline
## 1 & (1,1,1,0,0,1) & 4968.8 & 4987.52 \\
##   2 & (2,1,1,0,0,1) & 4964.1 & 4986.56 \\
##   3 & (0,1,2,0,0,1) & 4964.26 & 4982.98 \\
##   4 & (0,1,1,0,0,0) & 4974.16 & 4985.39 \\
##    \hline
## \end{tabular}
## \end{table}
```

```
#xtable(arima_total_4$ttable)
xtable(arima_sindi_3$ttable, label = "parámetros_sindi_011001", caption = "Parámetros del modelo (0,1,1
```

```
## % latex table generated in R 3.5.1 by xtable 1.8-3 package
## % Sun Nov 11 14:00:55 2018
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & SE & t.value & p.value \\
##   \hline
## ma1 & 0.06 & 0.06 & 0.99 & 0.32 \\
##   ma2 & 0.16 & 0.05 & 2.91 & 0.00 \\
##   sma1 & 0.15 & 0.06 & 2.57 & 0.01 \\
```

```
##   constant & 64.94 & 53.21 & 1.22 & 0.22 \\
##    \hline
## \end{tabular}
## \caption{Parámetros del modelo (0,1,1,0,0,1)}
## \label{parámetros_sindi_011001}
## \end{table}
```

Población condenada

```
### condenada 111001
png(file = 'Arima_conde_111001.png', height = 750, width = 500, res = 85)
arima_conde_1 <- sarima(ts_conde,1,1,1,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
### condenado 211001
png(file = 'Arima_conde_211001.png', height = 750, width = 500, res = 85)
arima_conde_2 <- sarima(ts_conde,2,1,1,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
auto.arima(ts_conde, ic = "aic")
```

```
## Series: ts_conde
## ARIMA(1,1,1)(0,0,1)[12] with drift
##
## Coefficients:
##          ar1      ma1    sma1     drift
##       0.8823  -0.7293  0.0935  198.6161
## s.e.  0.0641   0.0935  0.0604   86.2840
##
## sigma^2 estimated as 384640:  log likelihood=-2447.04
## AIC=4904.08   AICc=4904.27   BIC=4922.79
```

```
### condenado Auto Arima
png(file = 'Arima_conde_012001.png', height = 750, width = 500, res = 85)
arima_conde_3 <- sarima(ts_conde,0,1,2,0,0,1, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
### Total 121002
png(file = 'Arima_conde_011000.png', height = 750, width = 500, res = 85)
arima_conde_4 <- sarima(ts_conde,0,1,1,0,0,0, S = 12, details = FALSE)
dev.off()
```

```
## pdf
##   2
```

```
AIC_conde <- AIC(arima_conde_1$fit,arima_conde_2$fit, arima_conde_3$fit,arima_conde_4$fit)
BIC_conde <- BIC(arima_conde_1$fit,arima_conde_2$fit, arima_conde_3$fit,arima_conde_4$fit)
```

```
Indicadores_df_conde <- cbind(c("(1,1,1,0,0,1)","(2,1,1,0,0,1)","(0,1,1,0,0,1)","(0,1,1,0,0,0)"),round(A

colnames(Indicadores_df_conde) <- c("ORDEN","AIC","BIC")
xtable(Indicadores_df_conde)
```

```
## % latex table generated in R 3.5.1 by xtable 1.8-3 package
## % Sun Nov 11 14:00:56 2018
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlll}
##   \hline
##  & ORDEN & AIC & BIC \\
##   \hline
## 1 & (1,1,1,0,0,1) & 4904.08 & 4922.79 \\
##   2 & (2,1,1,0,0,1) & 4906 & 4928.46 \\
##   3 & (0,1,1,0,0,1) & 4914.84 & 4933.55 \\
##   4 & (0,1,1,0,0,0) & 4925.62 & 4936.85 \\
##    \hline
## \end{tabular}
## \end{table}
```

```
#xtable(arima_total_4$ttable)
xtable(arima_conde_1$ttable, label = "parámetros_conde_111001", caption = "Parámetros del modelo (1,1,1
```

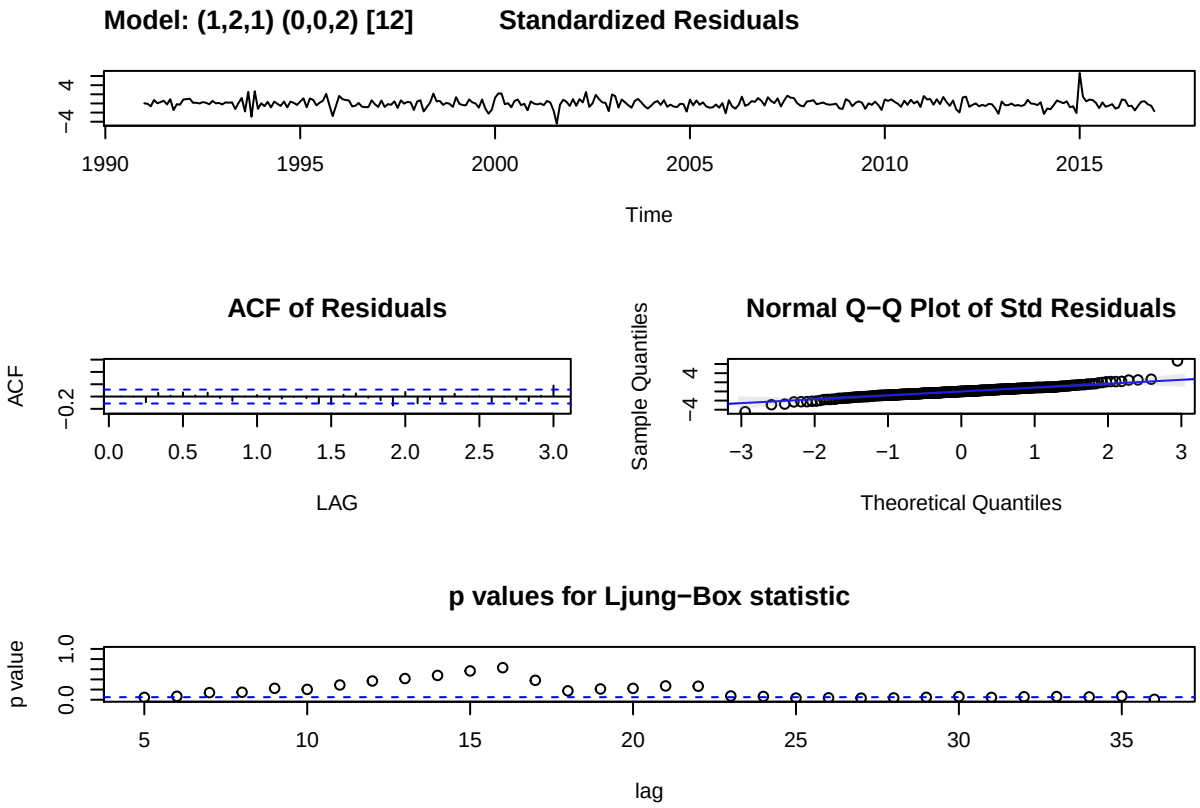
```
## % latex table generated in R 3.5.1 by xtable 1.8-3 package
## % Sun Nov 11 14:00:56 2018
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & SE & t.value & p.value \\
##   \hline
## ar1 & 0.88 & 0.06 & 13.75 & 0.00 \\
##   ma1 & -0.73 & 0.09 & -7.80 & 0.00 \\
##   sma1 & 0.09 & 0.06 & 1.55 & 0.12 \\
##   constant & 198.62 & 86.28 & 2.30 & 0.02 \\
##    \hline
## \end{tabular}
## \caption{Parámetros del modelo (1,1,1,0,0,1)}
## \label{parámetros_conde_111001}
## \end{table}
```

Ajuste de errores

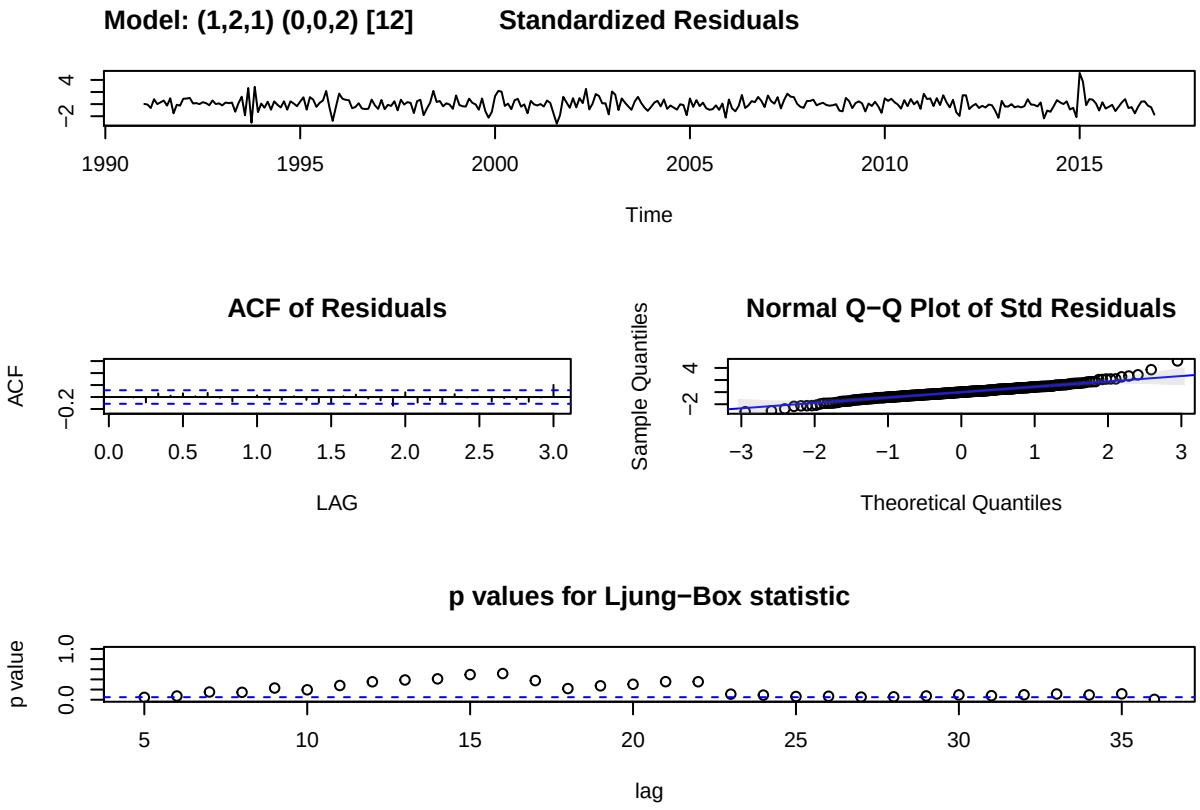
Población total

```
#se retira enero de 2017
ts_total <- ts(ts_total[1:312], start = c(1991,1), frequency = 12)
ts_sindi <- ts(ts_sindi[1:312], start = c(1991,1), frequency = 12)
ts_conde <- ts(ts_conde[1:312], start = c(1991,1), frequency = 12)
```

```
# Ejecutar mejor modelo
arima_total_4 <- sarima(ts_total,1,2,1,0,0,2, S = 12, details = FALSE)
```




```r
# Estraer residuales
residuales_arima_total_4 <- arima_total_4$fit$residuals
# Filtrar contra un umbral
residuales_out_arima_total_4 <- 1*(abs(residuales_arima_total_4)>(sqrt(arima_sindi_3$fit$sigma2)*3))

#Transformar a matriz Dummy
outliers <- t(as.data.frame(residuales_out_arima_total_4))
outliersdiag <-t(outliers)%*%as.matrix(outliers)*diag(dim(outliers)[2])
A <- outliersdiag
outliersdiagdummy <- as.matrix(A[,A[row(A)==col(A)]==1])

# Generar nuevo modelo
arima_total_4_dummy <- sarima(ts_total,1,2,1,0,0,2, S = 12, xreg = outliersdiagdummy , details = FALSE)
```




```
# seleccionar el mejor modelo con los nuevos datos
arima_select <- auto.arima(ts_total,ic = "aic", xreg = outliersdiagdummy)

png(file = 'Arima_total_211200Dummy.png', height = 750, width = 500, res = 85)
arima_total_4_dummy <- sarima(ts_total,2,1,1,2,0,0, S = 12, xreg = outliersdiagdummy , details = FALSE)
dev.off()
```

```
## pdf
##   2
```

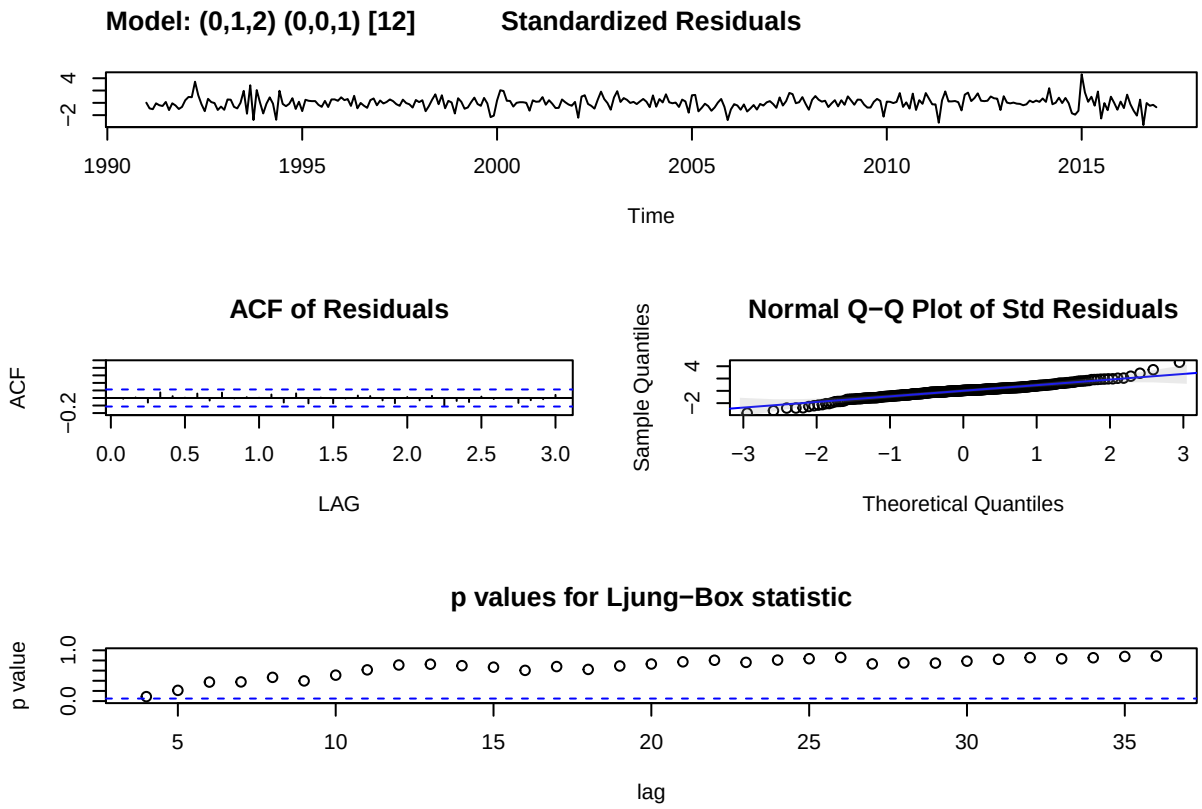
```
arima_total_4_dummystats <- stats::arima(x = ts_total, order = c(2, 1, 1), seasonal = list(order = c(2,
```

```
## initial  value 6.677430
## iter  10 value 6.372032
## iter  20 value 6.367628
## iter  30 value 6.355970
## iter  30 value 6.355970
## iter  30 value 6.355970
## final  value 6.355970
## converged
## initial  value 6.334529
## final  value 6.334097
## converged
```

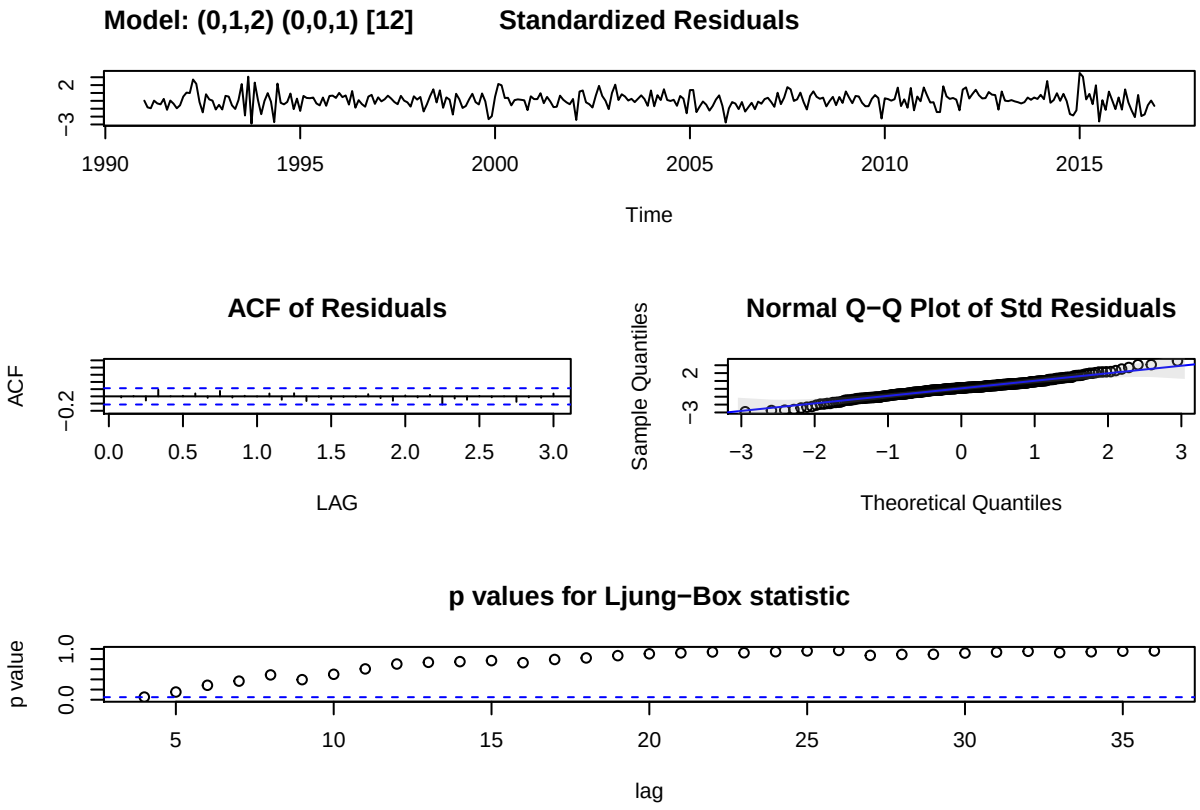
```
pry_dummy = matrix(0,36,dim(outliersdiagdummy)[2])

forecast_dummy_total <- predict(arima_total_4_dummystats, n.ahead = 36, newxreg = pry_dummy)
```

Población sindicada

```
# Ejecutar mejor modelo
arima_sindi_3 <- sarima(ts_sindi,0,1,2,0,0,1, S = 12, details = FALSE)
```




```
# Estraer residuales
residuales_arima_sindi_3 <- arima_sindi_3$fit$residuals
# Filtrar contra un umbral
residuales_out_arima_sindi_3 <- 1*(abs(residuales_arima_sindi_3)>(sqrt(arima_sindi_3$fit$sigma2)*3))

#Transformar a matriz Dummy
outliers <- t(as.data.frame(residuales_out_arima_sindi_3))
outliersdiag <-t(outliers)%*%as.matrix(outliers)*diag(dim(outliers)[2])
A <- outliersdiag
outliersdiagdummy <- as.matrix(A[,A[row(A)==col(A)]==1])

# Generar nuevo modelo
arima_sindi_3_dummy <- sarima(ts_sindi,0,1,2,0,0,1, S = 12, xreg = outliersdiagdummy , details = FALSE)
```




```
# seleccionar el mejor modelo con los nuevos datos
arima_select <- auto.arima(ts_sindi,ic = "aic", xreg = outliersdiagdummy)

png(file = 'Arima_sindi_012001Dummy.png', height = 750, width = 500, res = 85)
arima_sindi_3_dummy <- sarima(ts_sindi,0,1,2,0,0,1, S = 12, xreg = outliersdiagdummy , details = FALSE)
dev.off()
```

```
## pdf
##   2
```

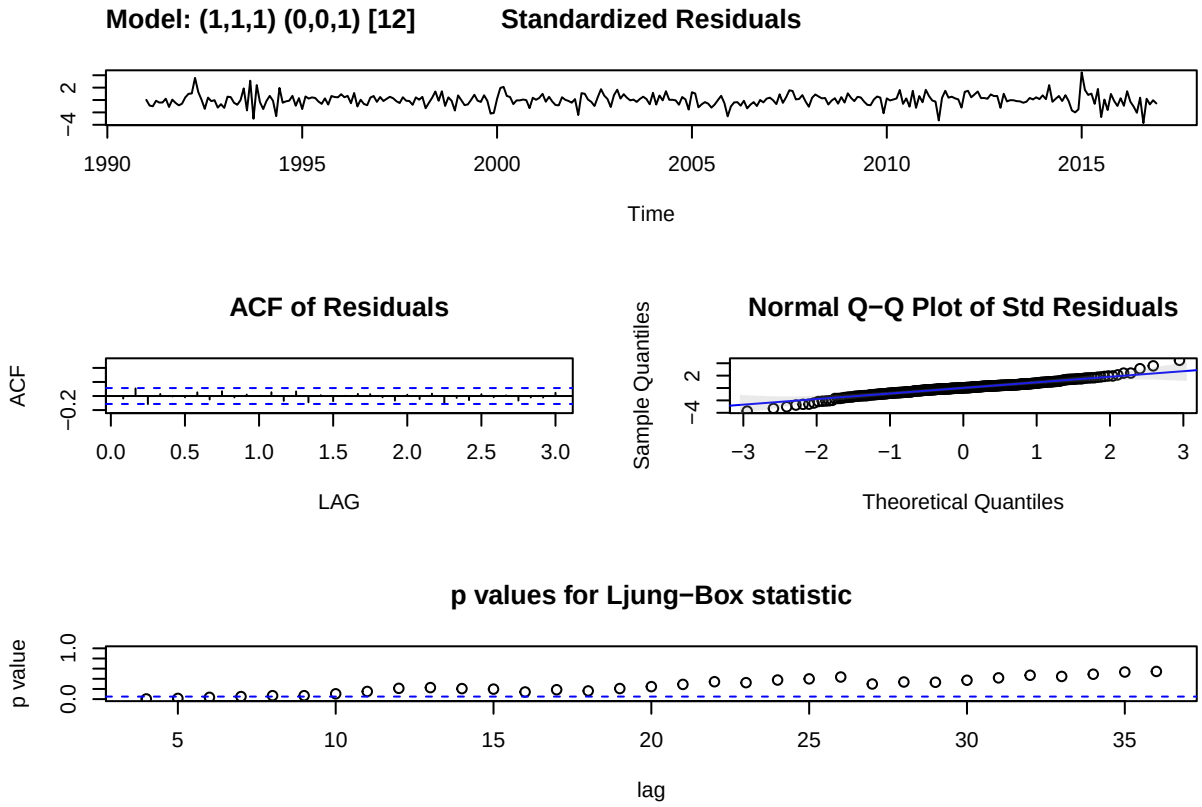
```
arima_sindi_3_dummystats <- stats::arima(x = ts_sindi, order = c(0, 1, 2), seasonal = list(order = c(0,
```

```
## initial  value 6.524097
## final  value 6.491939
## converged
## initial  value 6.492255
## final  value 6.492250
## converged
```

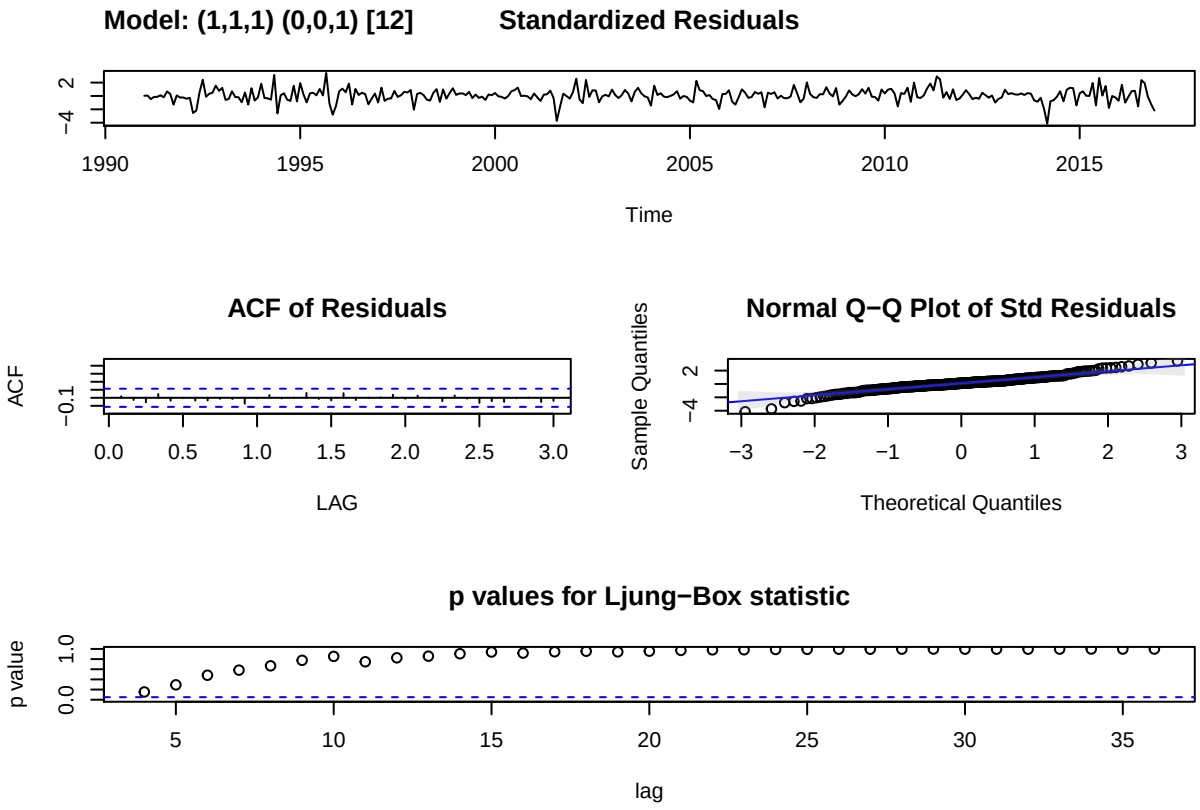
```
pry_dummy = matrix(0,36,dim(outliersdiagdummy)[2])

forecast_dummy_sindi <- predict(arima_sindi_3_dummystats, n.ahead = 36, newxreg = pry_dummy)
```

Población condenada

```r
# Ejecutar mejor modelo
arima_conde_1 <- sarima(ts_sindi,1,1,1,0,0,1, S = 12, details = FALSE)
```




```r
# Estraer residuales
residuales_arima_conde_1 <- arima_conde_1$fit$residuals
# Filtrar contra un umbral
residuales_out_arima_conde_1 <- 1*(abs(residuales_arima_sindi_3)>(sqrt(arima_conde_1$fit$sigma2)*3))

#Transformar a matriz Dummy
outliers <- t(as.data.frame(residuales_out_arima_conde_1))
outliersdiag <-t(outliers)%*%as.matrix(outliers)*diag(dim(outliers)[2])
A <- outliersdiag
outliersdiagdummy <- as.matrix(A[,A[row(A)==col(A)]==1])

# Generar nuevo modelo
arima_conde_1_dummy <- sarima(ts_conde,1,1,1,0,0,1, S = 12, xreg = outliersdiagdummy , details = FALSE)
```

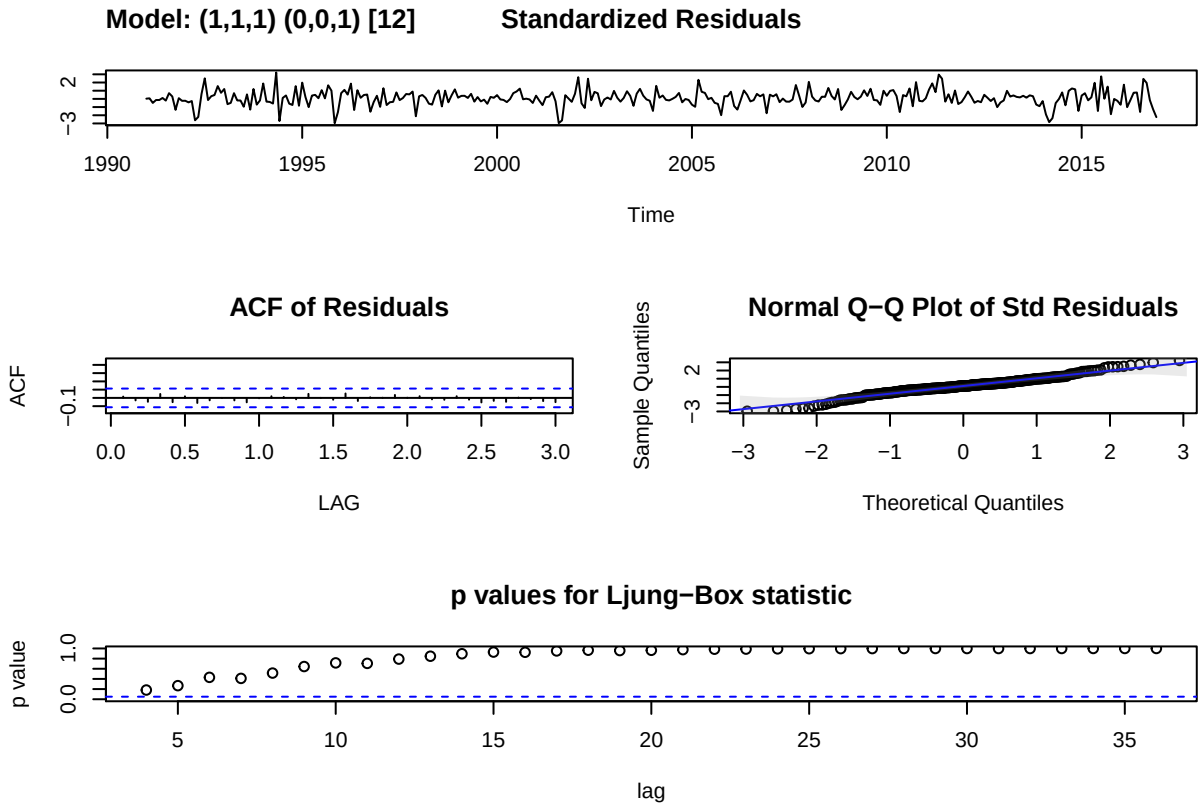



```
# seleccionar el mejor modelo con los nuevos datos
arima_select <- auto.arima(ts_conde,ic = "aic", xreg = outliersdiagdummy)

arima_conde_1_dummy <- sarima(ts_conde,1,1,1,0,0,1, S = 12, xreg = outliersdiagdummy , details = FALSE)

arima_conde_1_dummystats <- stats::arima(x = ts_conde, order = c(1, 1, 1), seasonal = list(order = c(0,
```

```
## initial  value 6.511261
## iter  10 value 6.394873
## final  value 6.393594
## converged
## initial  value 6.392852
## final  value 6.392828
## converged
```

```
residuales_arima_conde_11 <- arima_conde_1_dummy$fit$residuals
# Filtrar contra un umbral
residuales_out_arima_conde_1 <- (abs(residuales_arima_conde_11)>2000)*1

#Transformar a matriz Dummy
outliers <- t(as.data.frame(residuales_out_arima_conde_1))
outliersdiag <-t(outliers)%*%as.matrix(outliers)*diag(dim(outliers)[2])
A <- outliersdiag
outliersdiagdummy1 <- as.matrix(A[,A[row(A)==col(A)]==1])
outliersdiagdummys <- cbind(outliersdiagdummy1,outliersdiagdummy)

# Generar nuevo modelo
```

```
arima_conde_1_dummy1 <- sarima(ts_conde,1,1,1,0,0,1, S = 12, xreg = outliersdiagdummys , details = FALSE
```




```
# seleccionar el mejor modelo con los nuevos datos
arima_select <- auto.arima(ts_conde,ic = "aic", xreg = outliersdiagdummys)

png(file = 'Arima_conde_111001Dummy.png', height = 750, width = 500, res = 85)
arima_conde_1_dummy <- sarima(ts_conde,1,1,1,0,0,1, S = 12, xreg = outliersdiagdummys , details = FALSE
dev.off()
```

```
## pdf
##   2
```

```
arima_conde_1_dummystats <- stats::arima(x = ts_conde, order = c(1, 1, 1), seasonal = list(order = c(0,
```

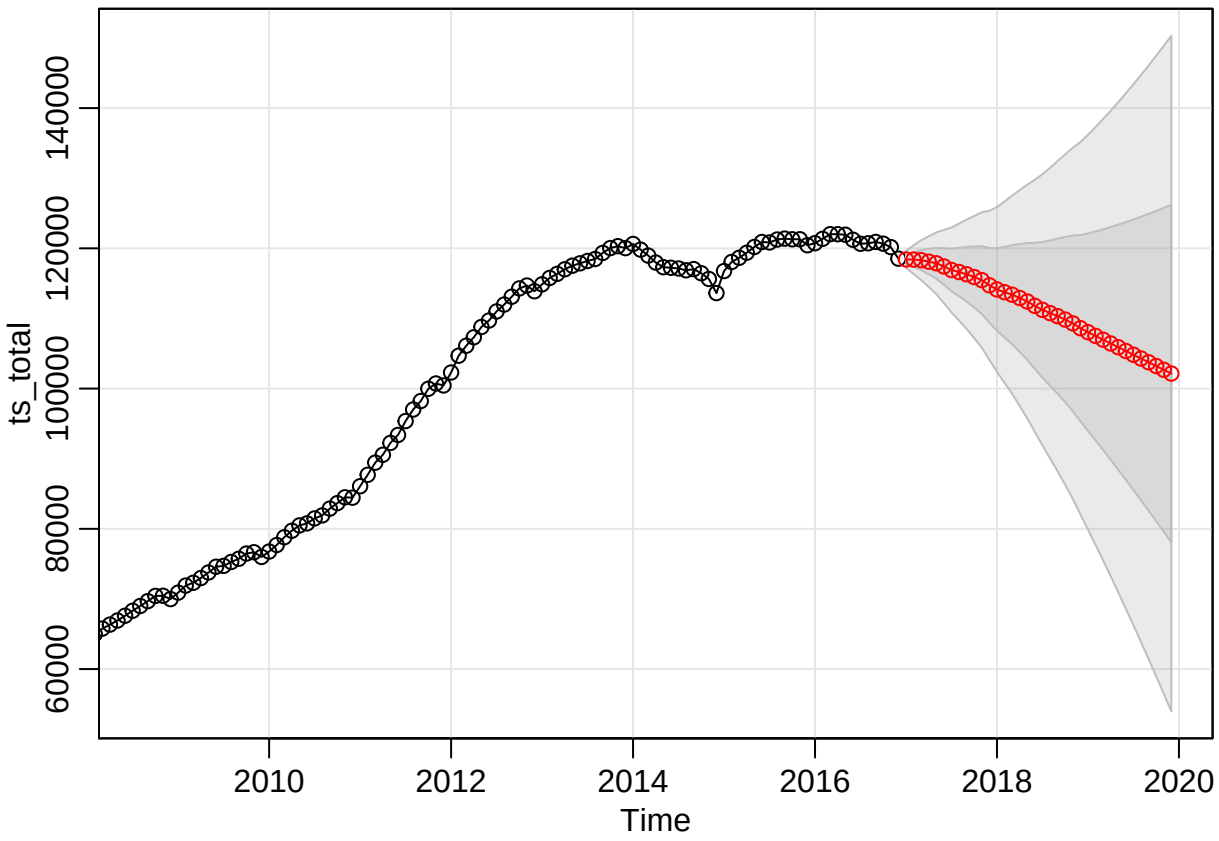
```
## initial  value 6.495115
## iter  10 value 6.378021
## final  value 6.370320
## converged
## initial  value 6.369572
## final  value 6.369550
## converged
```

```
# Pryectar
pry_dummy = matrix(0,36,dim(outliersdiagdummys)[2])

forecast_dummy_conde <- predict(arima_conde_1_dummystats, n.ahead = 36, newxreg = pry_dummy)
```

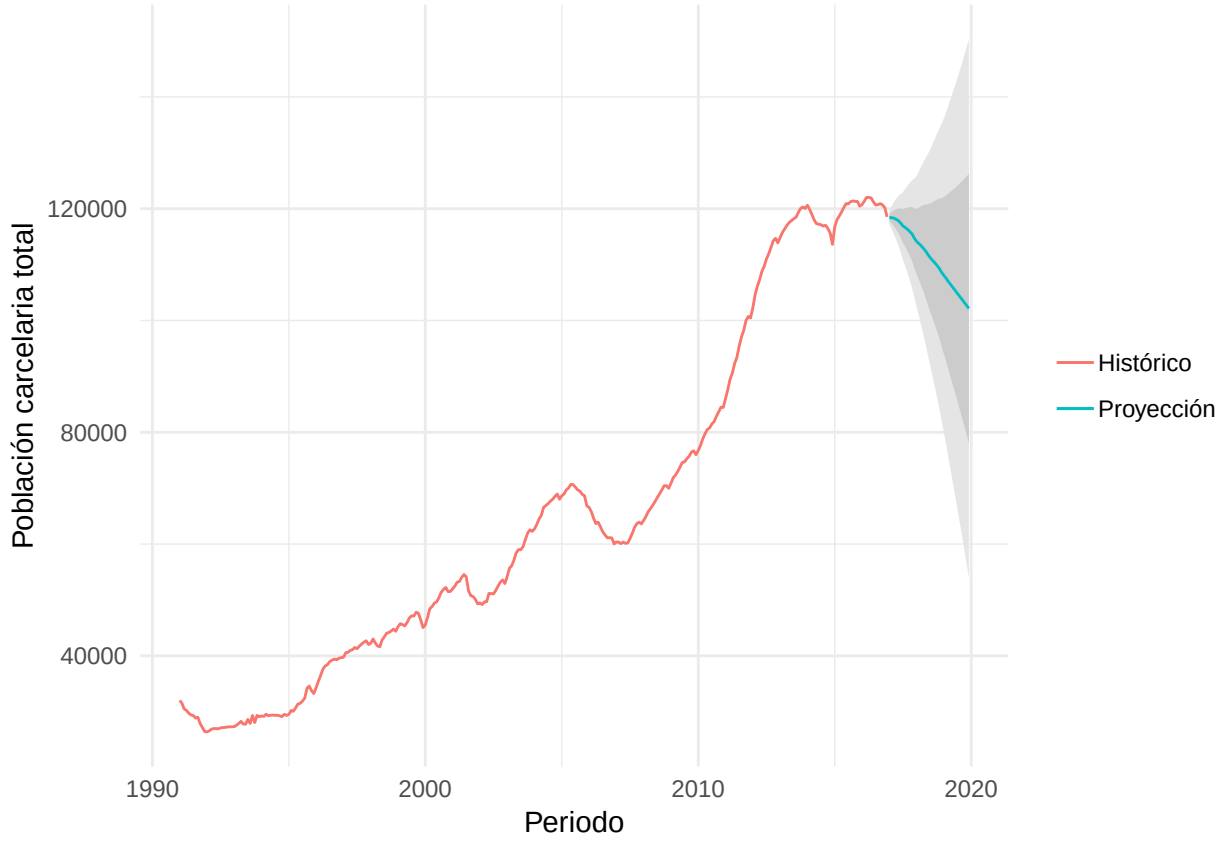
Proyecciones

Población total

```
# Proyección
forecast <- sarima.for(ts_total,n.ahead = 36, 1,2,1,0,0,2, S = 12)
```



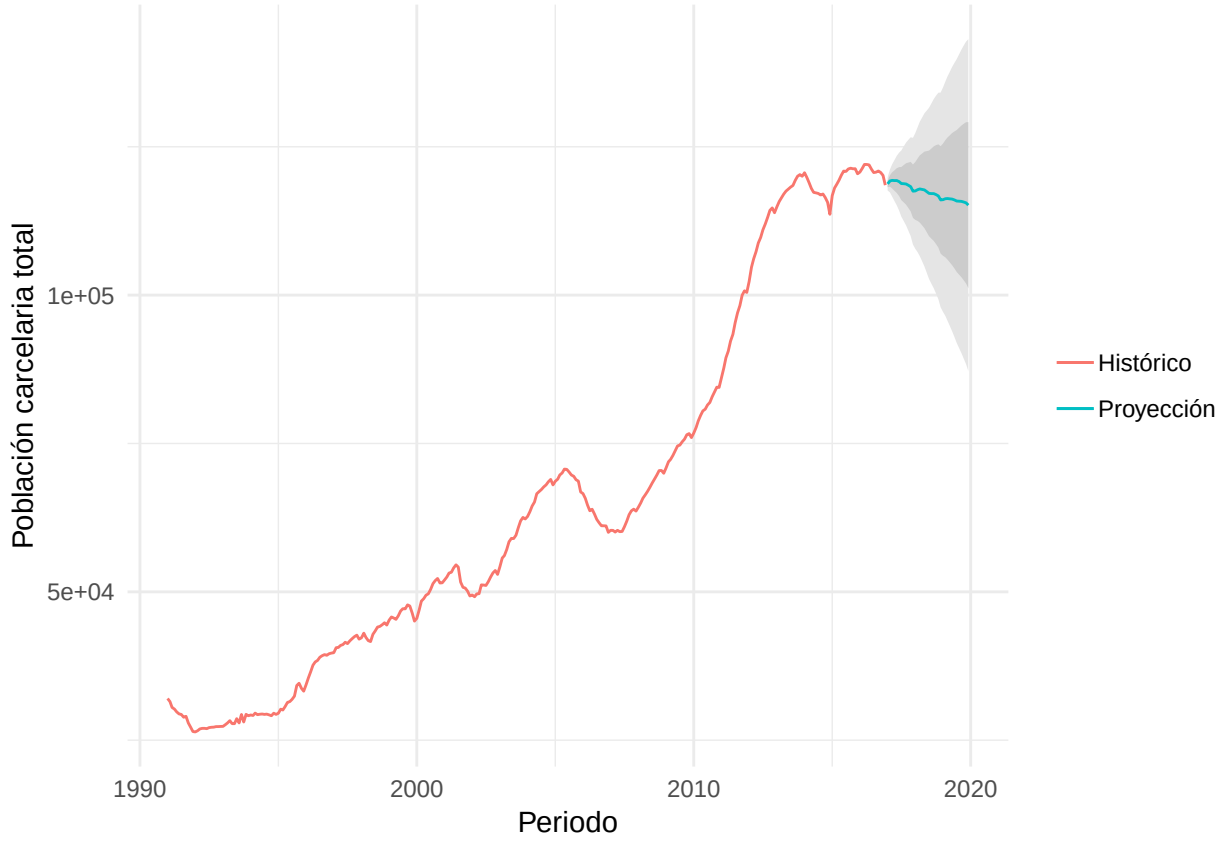

```
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_total), date=as.Date(as.yearmon(time(ts_total))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_total_4
```




```
ggsave("graf_pry_total_4.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
# Proyección
forecast <- forecast_dummy_total
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_total), date=as.Date(as.yearmon(time(ts_total))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_total_4
```

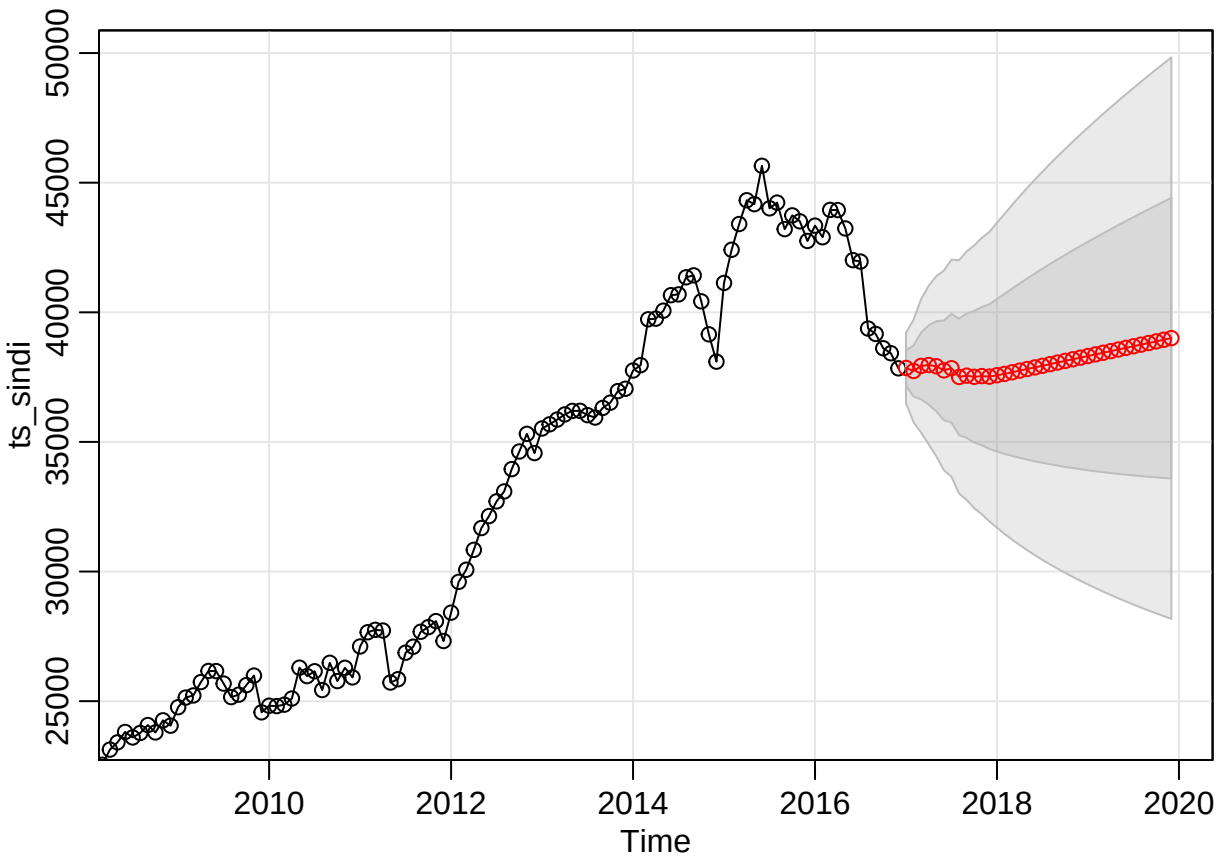



```
ggsave("graf_pry_total_4_dummy.png")
```

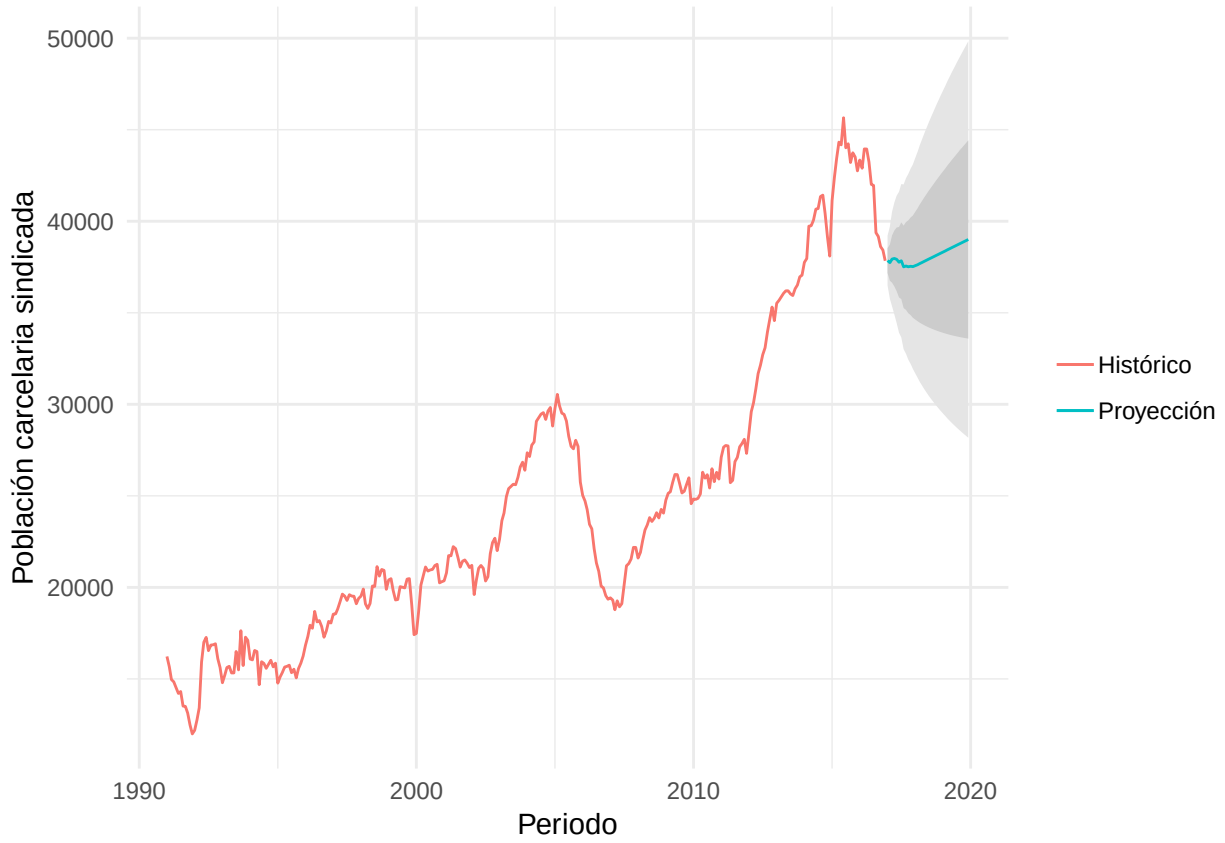
```
## Saving 6.5 x 4.5 in image
```

Población sindicada

```
# Proyección debe usar versión 1.8
forecast <- sarima.for(ts_sindi,n.ahead = 36, 0,1,2,0,0,1, S = 12 )
```



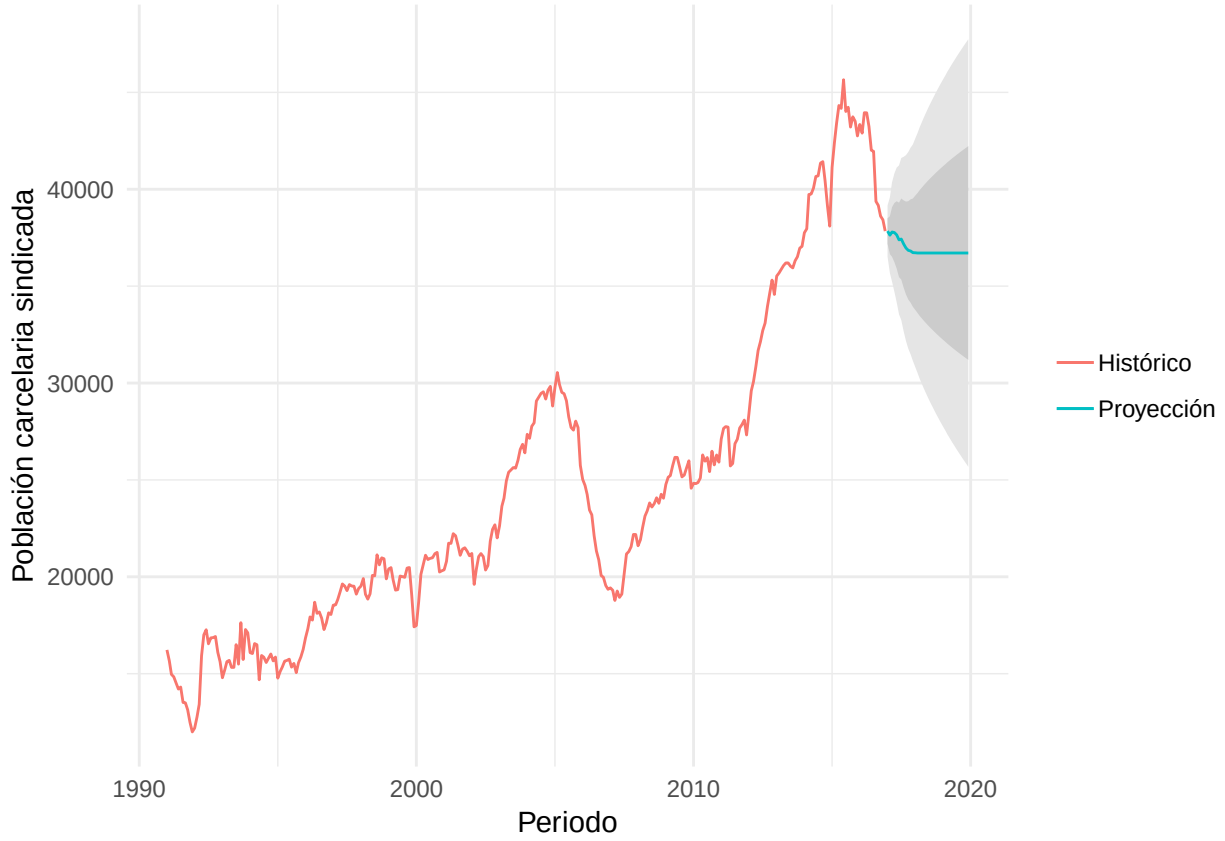

```
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_sindi), date=as.Date(as.yearmon(time(ts_sindi))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_total_3
```




```
ggsave("graf_pry_sindi_3.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
# Proyección debe usar versión 1.8
forecast <- forecast_dummy_sindi
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_sindi), date=as.Date(as.yearmon(time(ts_sindi))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_total_3
```

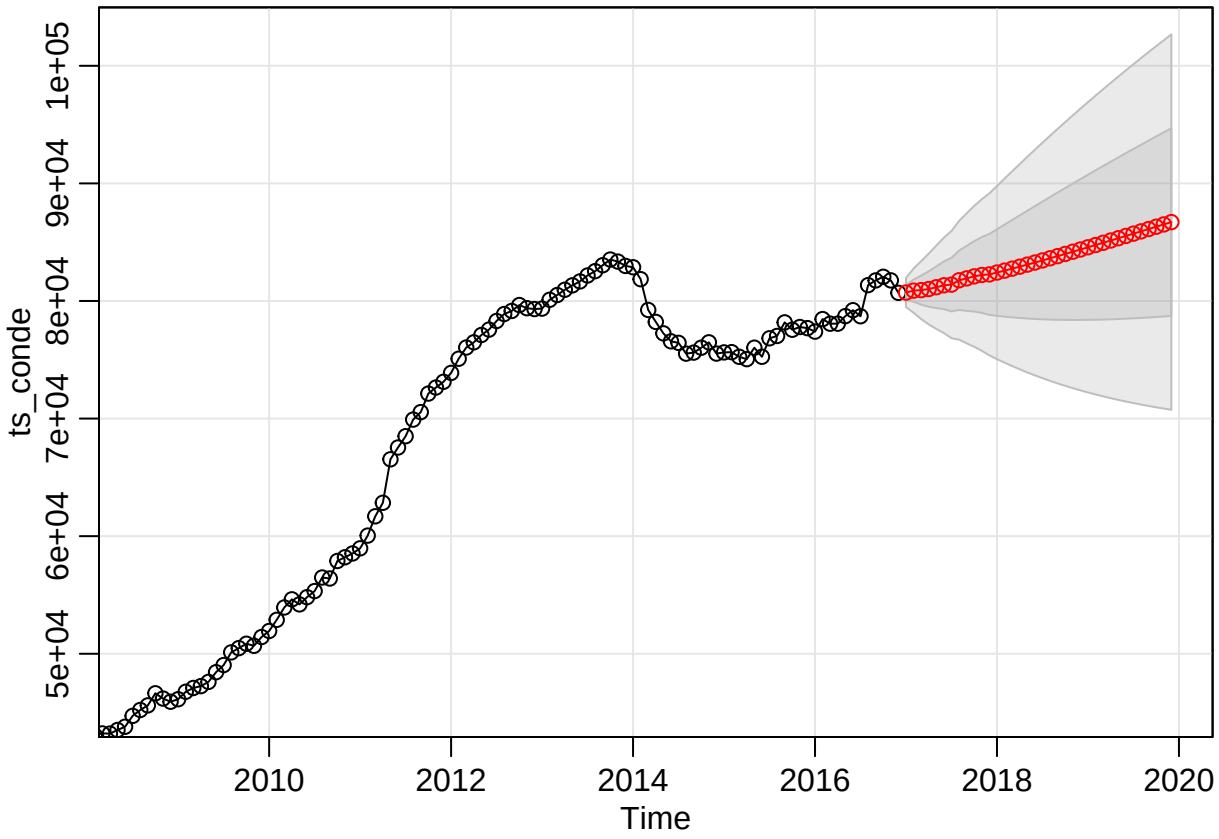



```
ggsave("graf_pry_sindi_3_dummy.png")
```

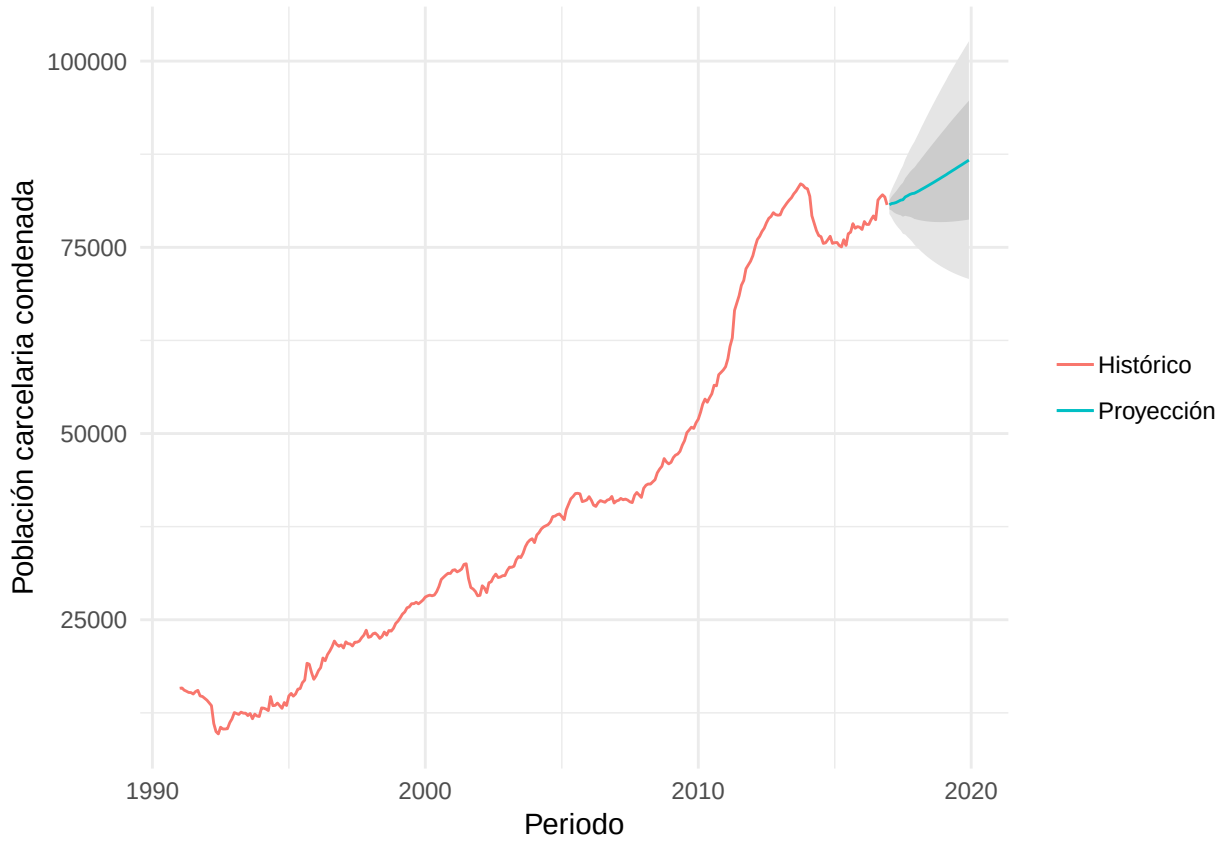
```
## Saving 6.5 x 4.5 in image
```

Población condenada

```
# Proyección
forecast <- sarima.for(ts_conde,n.ahead = 36, 1,1,1,0,0,1, S = 12)
```



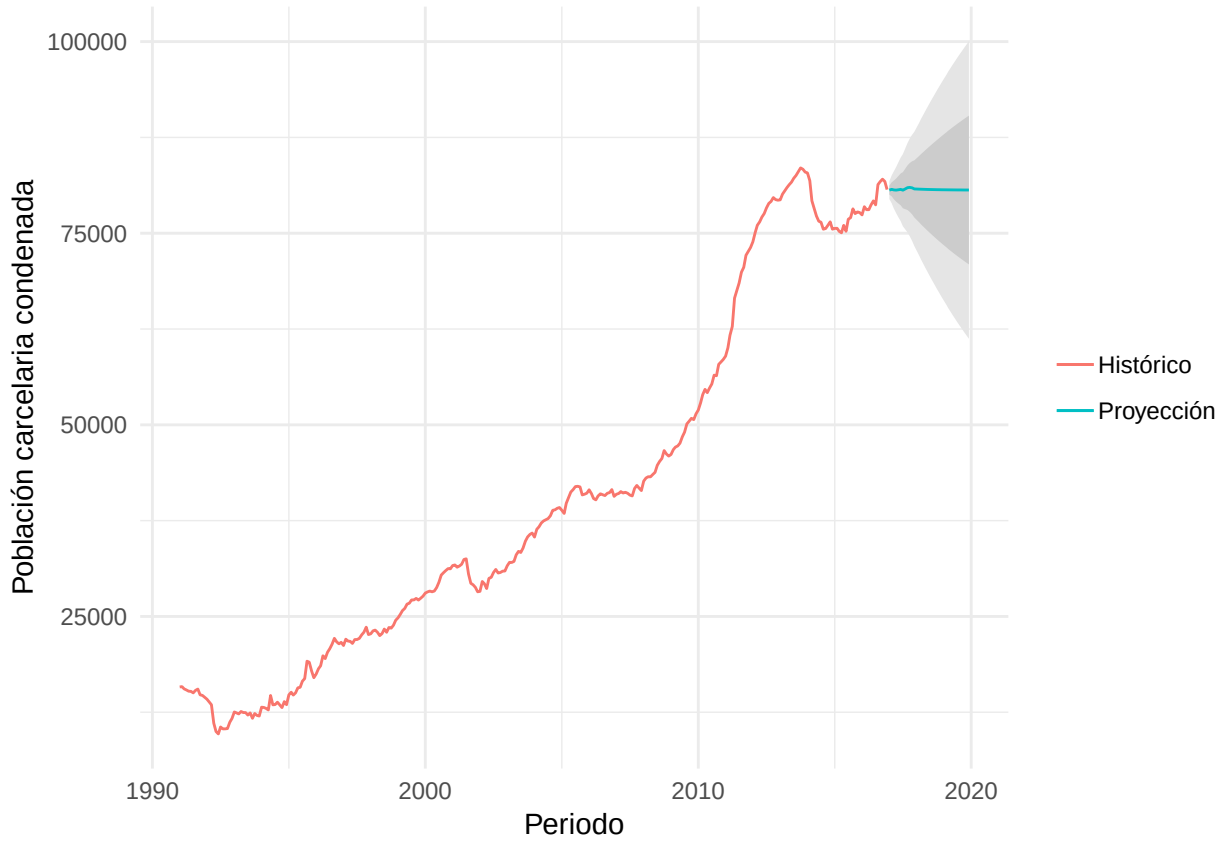

```
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_conde), date=as.Date(as.yearmon(time(ts_conde))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_conde_1
```




```
ggsave("graf_pry_conde_1.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
# Proyección
forecast <- forecast_dummy_conde
# data frame
forecast_frame <- data.frame(Proyección=as.matrix(forecast$pred),Error=as.matrix(forecast$se), date=as.
#data histórica
hist_frame <- data.frame(Historico=as.matrix(ts_conde), date=as.Date(as.yearmon(time(ts_conde))))
# ggplot
ggplot() + geom_ribbon(data = forecast_frame, aes(x = date, ymin = Proyección - 2*Error, ymax = Proyecc
graf_pry_conde_1
```




```
ggsave("graf_pry_conde_1_dummy.png")
```

```
## Saving 6.5 x 4.5 in image
```

Método de ratio Correlation.

```
# url <- "https://www.dropbox.com/s/94oblxtlc8anlk1/PRY_POB_NAL_DANE_EDAD.csv?dl=1"
# destfile <- "POB_NAL_DAN_EDAD.csv"
# curl_download(url, destfile)
```

```
# pob_nal_dane_edad = read.csv("POB_NAL_DAN_EDAD.csv",header = TRUE,sep = ",",stringsAsFactors=FALSE)
#
# pob_nal_dane_edad %<>% gather(Rango_edad, Poblacion, -Año, -Grupos.de.edad) %>% spread(Grupos.de.edad
#
# pob_nal_dane_edad %<>% dplyr::filter(Año>1990)
#
# pob_nal_dane_edad$Rango_edad<- gsub("X","",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("\\.","-",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("0-4","00-04",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("400-044","40-44",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("5-9","05-09",pob_nal_dane_edad$Rango_edad)
#
# pob_nal_dane_edad %>% ggplot(aes(x=Año, y = Total/1000000, fill = Rango_edad)) + geom_area(stat="iden
#
```

```
# pob_nal_dane_edad %>% ggplot(aes(x=Año, y = Total/1000000, colour = Rango_edad)) + geom_line() + them
#
# ### Crear columna de participación
#
# pob_nal_dane_edad %<>% group_by(Año) %>% summarise(Totales_año = sum(Total)) %>% right_join(pob_nal_d
#
# pob_nal_dane_edad %>% ggplot(aes(x=Año, y = Participacion, fill = Rango_edad)) + geom_area(stat="iden
#
# pob_nal_dane_edad %>% ggplot(aes(x=Año, y = Participacion, colour = Rango_edad)) + geom_line() + them
#
# # Generar gráfica
# #png(file = 'POB_NAL_EDAD.png', height = 850, width = 600, res = 85)
# POB_NAL_EDAD <- grid.arrange(pob_total_edad_stack, pob_total_edad_stack_part,pob_total_edad, pob_tota
# POB_NAL_EDAD
# #dev.off()
```

```
# # Cargar datos
# pob_nal_dane_edad = read.csv("POB_NAL_DAN_EDAD.csv",header = TRUE,sep = ",",stringsAsFactors=FALSE)
#
# # cambiar estructura donde el genero es columna y los gruposde edad es fila
# pob_nal_dane_edad %<>% gather(Rango_edad, Poblacion, -Año, -Grupos.de.edad) %>% spread(Grupos.de.edad
#
# pob_nal_dane_edad %<>% dplyr::filter(Año>1990)
#
# # Corregir titulos de rango etario
# pob_nal_dane_edad$Rango_edad<- gsub("X","",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("\\.","-",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("0-4","00-04",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("400-044","40-44",pob_nal_dane_edad$Rango_edad)
# pob_nal_dane_edad$Rango_edad<- gsub("5-9","05-09",pob_nal_dane_edad$Rango_edad)
#
# # Población DANE por edad
# pob_nal_dane_edad %>% dplyr::select(Año, Rango_edad, Total) %>% tidyr::spread(Rango_edad, Total) -> p
#
# # Acumular rangos etarios que se trabajan acumulados
# Menor_14 <-apply(pob_dane_edad[,2:4],1,sum)
# Mayor_60 <- apply(pob_dane_edad[,16:18],1,sum)
#
# # Crear arreglo de regresores
# Regresores <- cbind(Menor_14,pob_dane_edad[,5:15],Mayor_60)
#
# # Calcular población total
# Pob_total <- apply(Regresores,1,sum)
#
# # Población total en mayo de cada año
# ppl_sitjur %>% dplyr::filter(categoria =="total", mes == 5, !is.na(valor)) %>% dplyr::select(valor, a
#
# # Calcular tasa de encarcelamiento
# registros <- dim(pob_inpec_total)[1]
# tasa_encar <- pob_inpec_total[,1]/Pob_total[1:registros]
#
#
#
# # Guardar modelo de regresión
```

```
# tasa_regresores <- cbind(tasa_encar = pob_inpec_total[,1],Regresores[1:registros,])
#
# # Salvar
# write.table(tasa_regresores,"tasa_regresores.csv",sep=",",row.names = FALSE)
```

```
# tasa_regresores <- read.table("tasa_regresores.csv", sep = ",", header = TRUE)
#
# ddist <- datadist(tasa_regresores)
# options(datadst = "ddist")
#
# reg <- lm(tasa_encar ~ -1 + X15.19 + X20.24  + X20.24 + X25.29 + X30.34 + X35.39 + X40.44 + X45.49 + X
# summary(reg)
#
# step(reg,direction="backward")
```