ANÁLISIS Y PREDICCIÓN DE SERIES TEMPORALES

3/4/2021

1. Introducción: Presentación de la serie a analizar.

```
# 1)
METRO <- read_excel("Metro.xlsx")
```

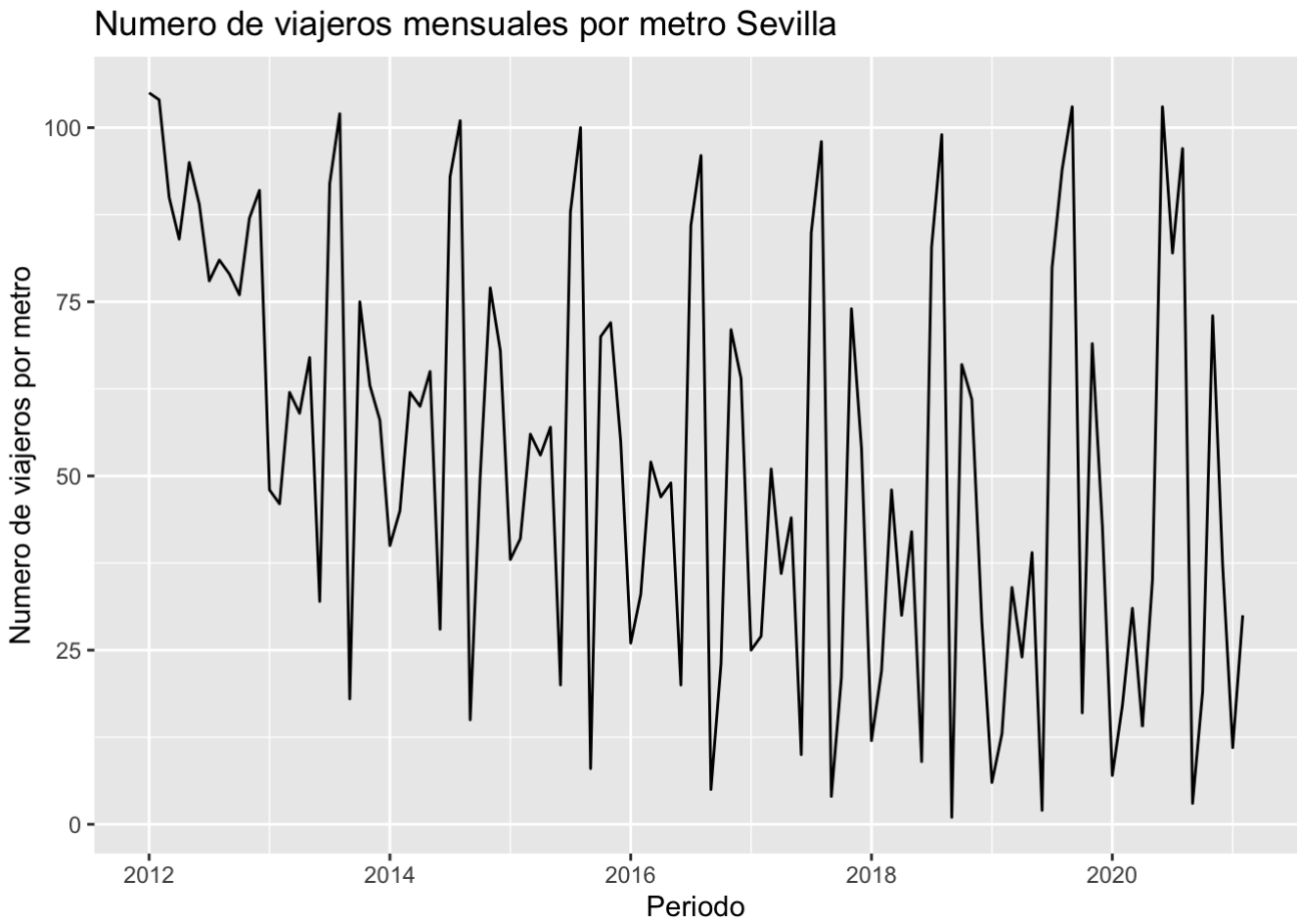
```
## New names:
## * `` -> ...1
```

```
metro<-ts(METRO[,-1],start = c(2012,1),frequency = 12)
```

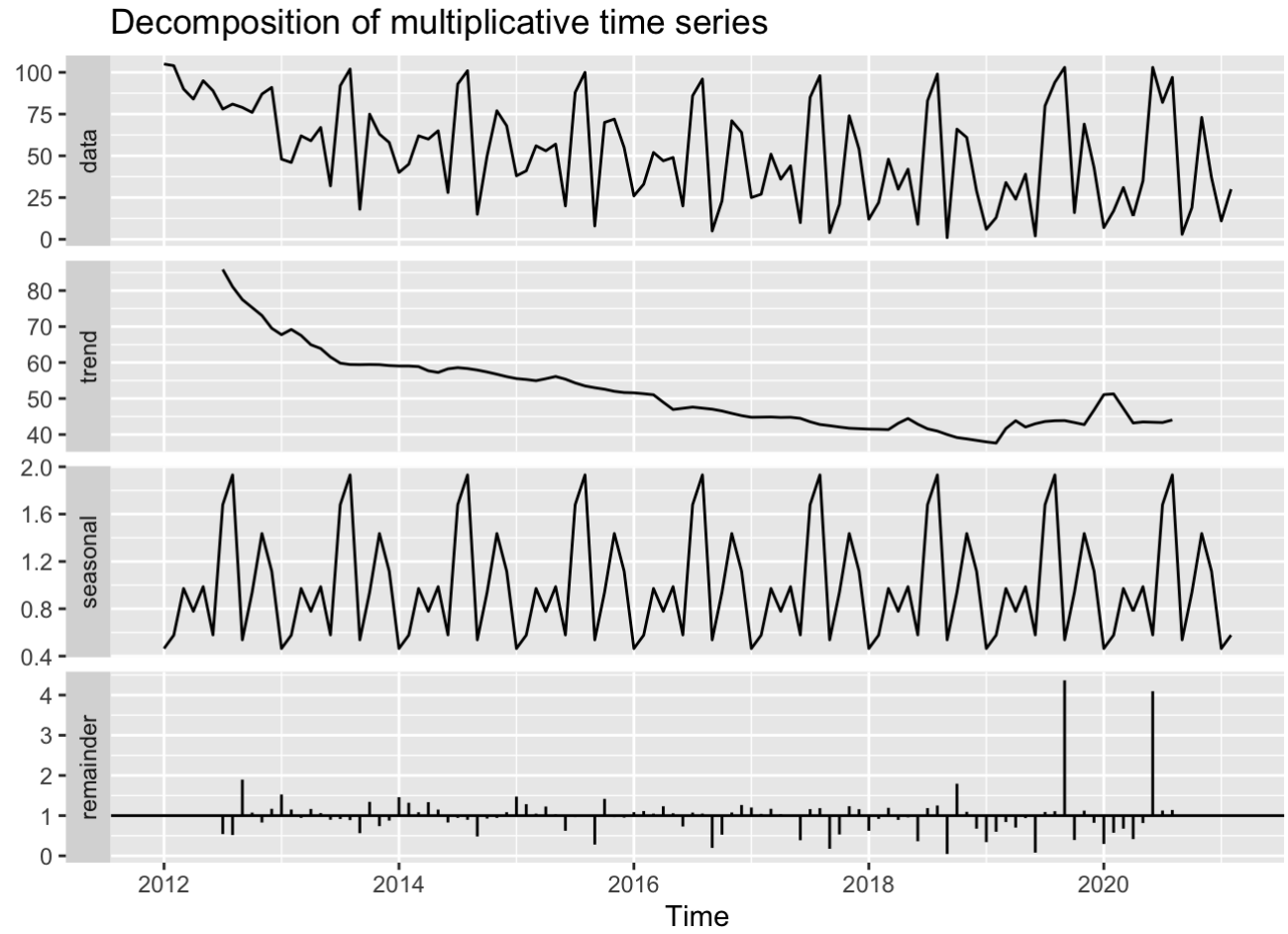
Los datos de la serie a analizar se organizan mensualmente. El número de valores es 108. He descargado los datos de 'transporte urbano en metro' de Sevilla, de 2012 a 2020, en la página web del INE (Instituto Nacional de Estadística).

2. Representación gráfica y descomposición estacional (si tuviera comportamiento estacional).

```
# 2.1)
autoplot(metro)+ ggtitle("Numero de viajeros mensuales por metro Sevilla") + xlab("Periodo") + ylab("Numero de viajeros por metro")
```




```
# 2.2)
metro_Comp<- decompose(metro,type=c("multiplicative"))
autoplot(metro_Comp,ts.colour = "blue")
```




```
# 2.3)
knitr::kable(metro_Comp$figure, digits =2,caption = "Coef Estacionalidad")
```

Coef Estacionalidad

x
0.46
0.58
0.97
0.78
0.99
0.58
1.68
1.93
0.54
0.94
1.44
1.12

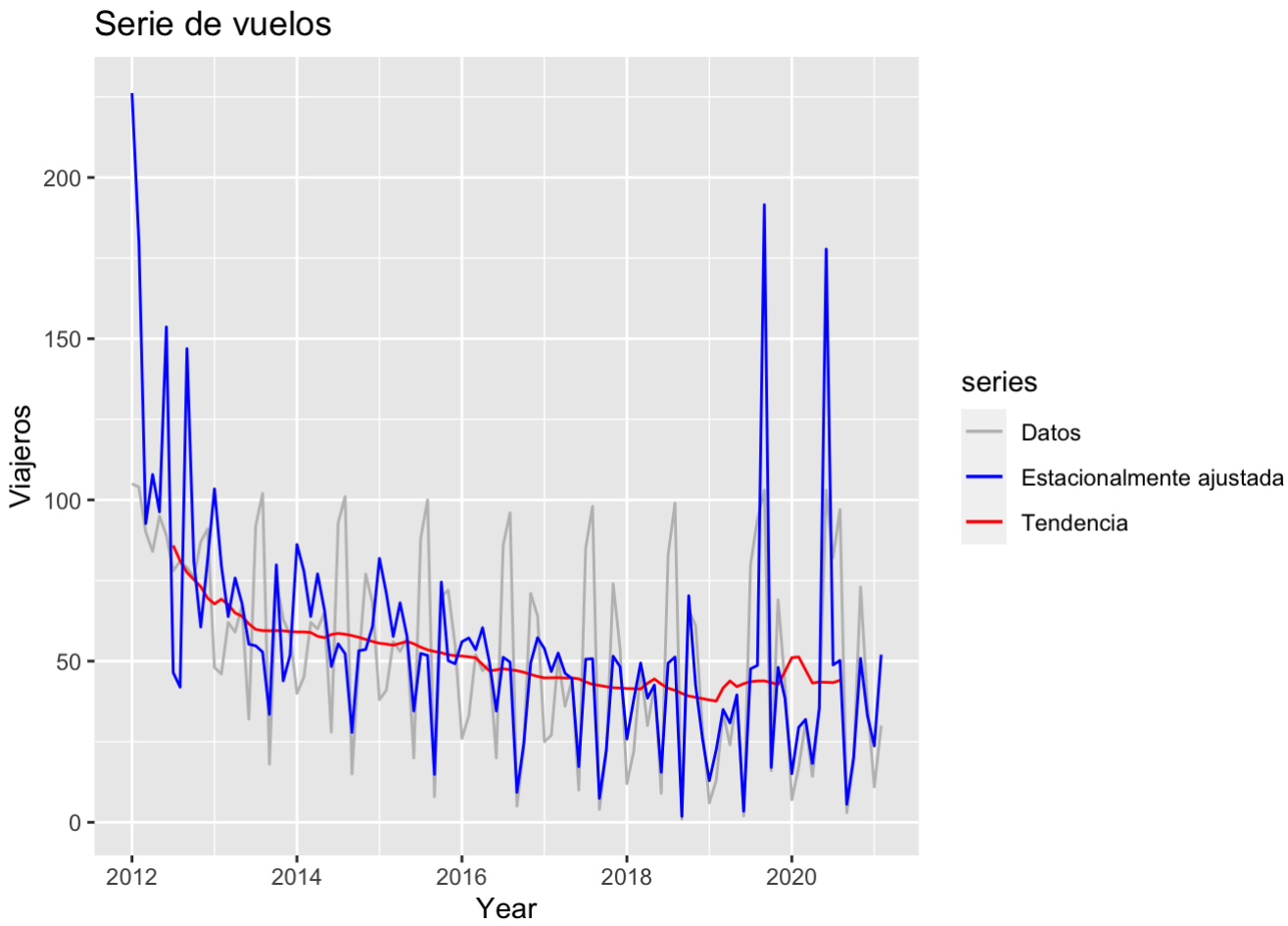
```
# 2.4)
print(metro_Comp)
```

```
## $x
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2012 105 104  90  84  95  89  78  81  79  76  87  91
## 2013  48  46  62  59  67  32  92 102  18  75  63  58
## 2014  40  45  62  60  65  28  93 101  15  50  77  68
## 2015  38  41  56  53  57  20  88 100   8  70  72  55
## 2016  26  33  52  47  49  20  86  96   5  23  71  64
## 2017  25  27  51  36  44  10  85  98   4  21  74  54
## 2018  12  22  48  30  42   9  83  99   1  66  61  29
## 2019   6  13  34  24  39   2  80  94 103  16  69  43
## 2020   7  17  31  14  35 103  82  97   3  19  73  37
## 2021  11  30
##
## $seasonal
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2012 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2013 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2014 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2015 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2016 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2017 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2018 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2019 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2020 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
## 2021 0.4641746 0.5771365
##            Aug       Sep       Oct       Nov       Dec
## 2012 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2013 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2014 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2015 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2016 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2017 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2018 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2019 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2020 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
## 2021
##
## $trend
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2012       NA       NA       NA       NA       NA       NA 85.87500 81.08333
## 2013 67.75000 69.20833 67.54167 64.95833 63.91667 61.54167 59.83333 59.45833
## 2014 59.04167 59.04167 58.87500 57.70833 57.25000 58.25000 58.58333 58.33333
## 2015 55.54167 55.29167 54.95833 55.50000 56.12500 55.37500 54.33333 53.50000
## 2016 51.58333 51.33333 51.04167 48.95833 46.95833 47.29167 47.62500 47.33333
## 2017 44.79167 44.83333 44.87500 44.75000 44.79167 44.50000 43.54167 42.79167
## 2018 41.50000 41.45833 41.37500 43.12500 44.45833 42.87500 41.58333 40.95833
## 2019 37.95833 37.62500 41.66667 43.83333 42.08333 43.00000 43.62500 43.83333
## 2020 51.08333 51.29167 47.25000 43.20833 43.50000 43.41667 43.33333 44.04167
## 2021       NA       NA
##           Sep      Oct      Nov      Dec
## 2012 77.50000 75.29167 73.08333 69.54167
## 2013 59.41667 59.45833 59.41667 59.16667
## 2014 57.91667 57.37500 56.75000 56.08333
## 2015 53.00000 52.58333 52.00000 51.66667
## 2016 47.04167 46.54167 45.87500 45.25000
## 2017 42.45833 42.08333 41.75000 41.62500
## 2018 40.00000 39.16667 38.79167 38.37500
```

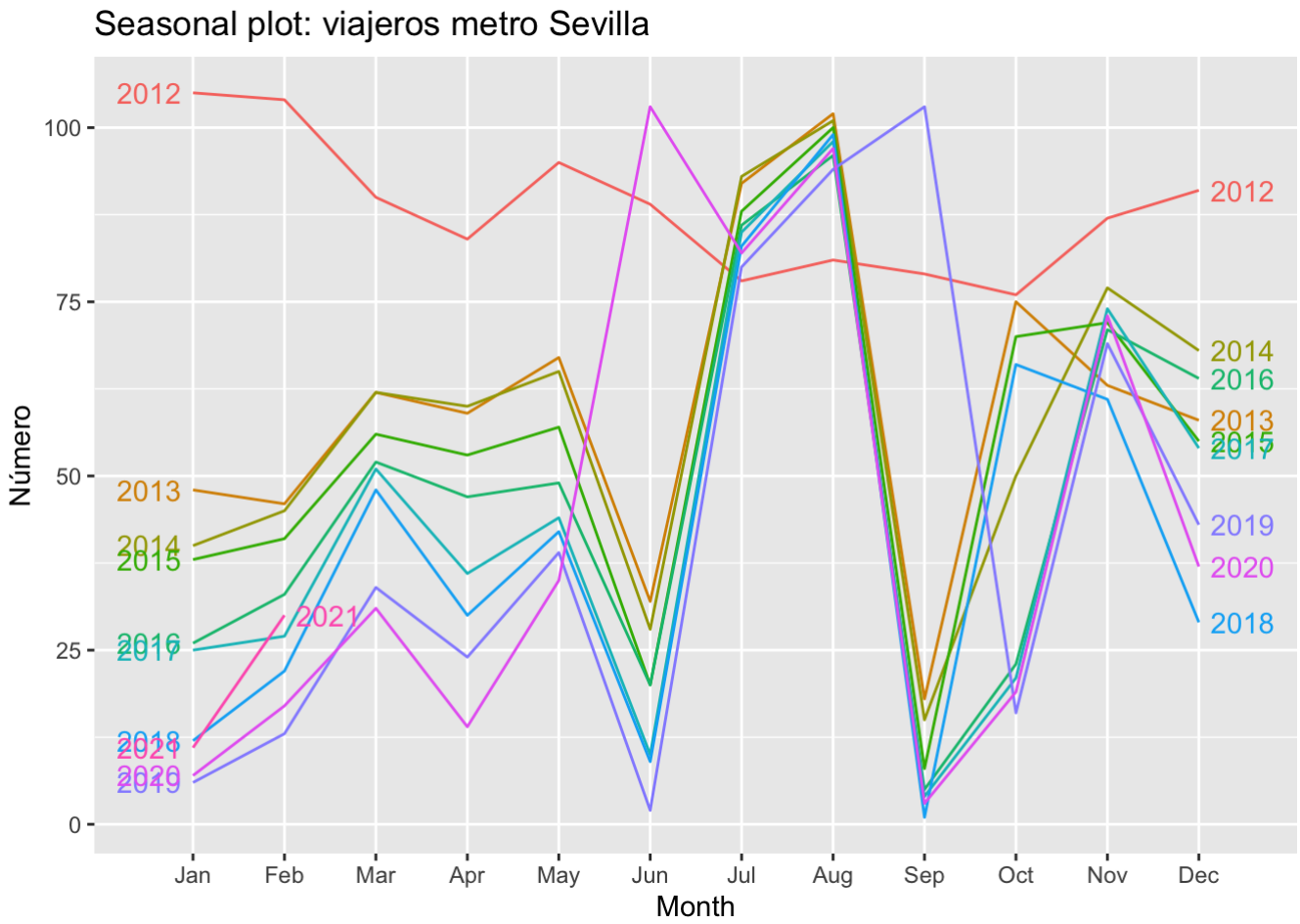
```
## 2019 43.87500 43.33333 42.75000 46.79167
## 2020       NA       NA       NA       NA
## 2021
##
## $random
##             Jan        Feb        Mar        Apr        May        Jun
## 2012         NA         NA         NA         NA         NA         NA
## 2013 1.52633754 1.15165112 0.94502759 1.16646172 1.06227880 0.89761302
## 2014 1.45955354 1.32061249 1.08413993 1.33526072 1.15057702 0.82979444
## 2015 1.47395198 1.28483001 1.04900859 1.22641158 1.02919183 0.62348308
## 2016 1.08588189 1.11387375 1.04882509 1.23289073 1.05745339 0.73005200
## 2017 1.20243451 1.04348024 1.17001192 1.03314866 0.99548170 0.38792557
## 2018 0.62294800 0.91945893 1.19433952 0.89339908 0.95735705 0.36236546
## 2019 0.34053579 0.59867116 0.84006856 0.70316962 0.93914425 0.08029157
## 2020 0.29521435 0.57427989 0.67543639 0.41611548 0.81537354 4.09533247
## 2021         NA         NA
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2012 0.54064963 0.51722945 1.89575721 1.07461527 0.82891227 1.17038774
## 2013 0.91523515 0.88821328 0.56340611 1.34287275 0.73831201 0.87676732
## 2014 0.94492411 0.89646719 0.48166493 0.92775570 0.94478399 1.08444757
## 2015 0.96406071 0.96777865 0.28071877 1.41721668 0.96413271 0.95210690
## 2016 1.07485906 1.05010799 0.19767176 0.52610476 1.07768028 1.26501261
## 2017 1.16198879 1.18575972 0.17520818 0.53124578 1.23419251 1.16030718
## 2018 1.18808319 1.25147662 0.04649405 1.79396370 1.09496203 0.67590098
## 2019 1.09154742 1.11033282 4.36593659 0.39308296 1.12388180 0.82192730
## 2020 1.12636673 1.14034906         NA         NA         NA         NA
## 2021
##
## $figure
##  [1] 0.4641746 0.5771365 0.9713493 0.7786578 0.9867842 0.5792841 1.6800103
##  [8] 1.9313909 0.5377033 0.9393202 1.4361253 1.1180637
##
## $type
## [1] "multiplicative"
##
## attr(,"class")
## [1] "decomposed.ts"
```

```
# 2.5)
autoplot(metro, series="Datos") +
autolayer(trendcycle(metro_Comp), series="Tendencia")+ autolayer(seasadj(metro_Comp),
series="Estacionalmente ajustada")+ xlab("Year") + ylab("Viajeros") +
ggtitle("Serie de vuelos") + scale_colour_manual(values=c("gray","blue","red"), break
s=c("Datos","Estacionalmente ajustada","Tendencia"))
```

```
## Warning: Removed 12 row(s) containing missing values (geom_path).
```




```
# 2.6)
ggseasonplot(metro, year.labels=TRUE, year.labels.left=TRUE) + ylab("Número") +
ggtitle("Seasonal plot: viajeros metro Sevilla")
```



Nos centramos en la serie viajeros en metro en Sevilla. En primer lugar, realizamos la descomposición estacional según el modelo multiplicativo. Después, representamos los componentes de la serie obtenidos.

En la gráfica que representa la estacionalidad, podemos ver que es la representación de los coeficientes estacionales, por lo que se repite de manera constante.

Según los coeficientes de estacionalidad observemos que el mayor es 1.93, que corresponde al mes de agosto. El menor de los coeficientes es el del mes de enero, 0.46.

Recordamos que para estudiar el comportamiento estacional resulta útil la representación gráfica de los valores de la serie, dibujando cada año en un color diferente. Como podemos ver, los años tienen la misma estructura, salvo en 2012, en la gráfica de los valores de la serie dibujado cada año en un color diferente, y salvo el año 2020, año de la pandemia. Por otro lado, se ve que va aumentando el número de viajeros cada año.

En conclusion, una serie temporal es el resultado de observar los valores de una variable a lo largo del tiempo en intervalos regulares. Por ello, la representación gráfica tiene un comportamiento estacional. De hecho, las fluctuaciones estacionales son aproximadamente constantes en el tiempo, salvo en el ggseasonplot.

3. Para comprobar la eficacia de los métodos de predicción que vamos a hacer en los siguientes apartados reservamos los últimos datos observados (un periodo en las series estacionales o aproximadamente

10 observaciones) para comparar con las predicciones realizadas por cada uno de los métodos.

```
# 3)
train <- window(metro, end=c(2018,12), frequency = 12)
```

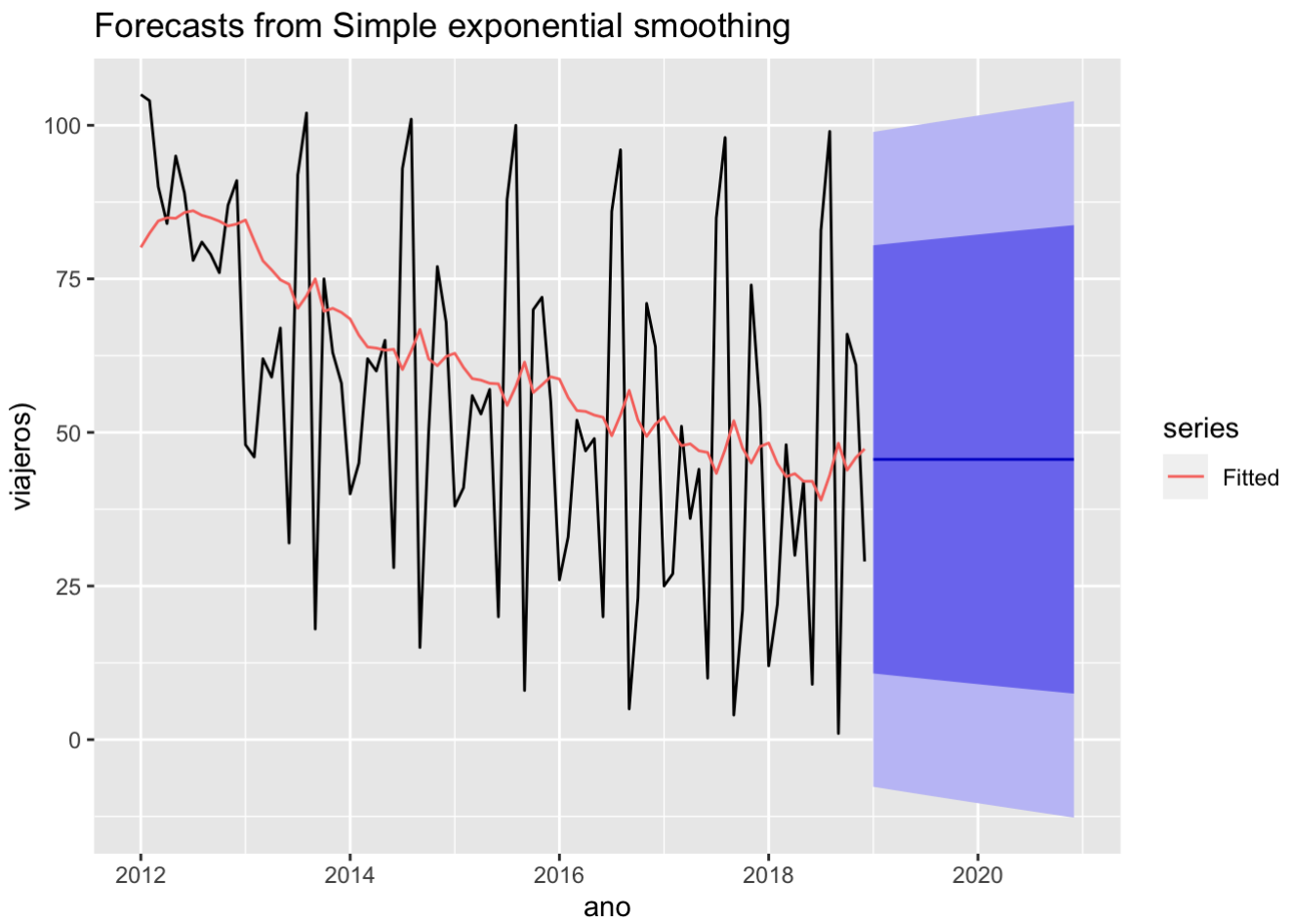
Reservaremos los datos de 2019 y 2020 para no utilizarlos en el entrenamiento.

4. Encontrar el modelo de suavizado exponencial más adecuado. Para dicho modelo, representar gráficamente la serie observada y la suavizada con las predicciones para un periodo que se considere adecuado.

```
# 4.1)
metro_s1=ses(train,alpha=NULL,h=24)
print(metro_s1)
```

```
##          Point Forecast     Lo 80    Hi 80      Lo 95     Hi 95
## Jan 2019        45.6175 10.773928 80.46106  -7.671142  98.90613
## Feb 2019        45.6175 10.624895 80.61010  -7.899068  99.13406
## Mar 2019        45.6175 10.476495 80.75850  -8.126028  99.36102
## Apr 2019        45.6175 10.328718 80.90627  -8.352033  99.58702
## May 2019        45.6175 10.181557 81.05343  -8.577095  99.81209
## Jun 2019        45.6175 10.035006 81.19999  -8.801227 100.03622
## Jul 2019        45.6175  9.889055 81.34594  -9.024439 100.25943
## Aug 2019        45.6175  9.743698 81.49129  -9.246744 100.48174
## Sep 2019        45.6175  9.598928 81.63606  -9.468151 100.70314
## Oct 2019        45.6175  9.454737 81.78025  -9.688672 100.92366
## Nov 2019        45.6175  9.311119 81.92387  -9.908317 101.14331
## Dec 2019        45.6175  9.168066 82.06693 -10.127096 101.36209
## Jan 2020        45.6175  9.025573 82.20942 -10.345020 101.58001
## Feb 2020        45.6175  8.883633 82.35136 -10.562099 101.79709
## Mar 2020        45.6175  8.742239 82.49275 -10.778343 102.01333
## Apr 2020        45.6175  8.601385 82.63361 -10.993760 102.22875
## May 2020        45.6175  8.461065 82.77393 -11.208361 102.44335
## Jun 2020        45.6175  8.321273 82.91372 -11.422154 102.65715
## Jul 2020        45.6175  8.182003 83.05299 -11.635149 102.87014
## Aug 2020        45.6175  8.043250 83.19174 -11.847355 103.08235
## Sep 2020        45.6175  7.905006 83.32999 -12.058780 103.29377
## Oct 2020        45.6175  7.767268 83.46772 -12.269432 103.50442
## Nov 2020        45.6175  7.630029 83.60496 -12.479321 103.71431
## Dec 2020        45.6175  7.493284 83.74171 -12.688454 103.92345
```

```
autoplot(metro_s1) + autolayer(fitted(metro_s1), series="Fitted") + ylab("viajeros")")
+ xlab("ano")
```

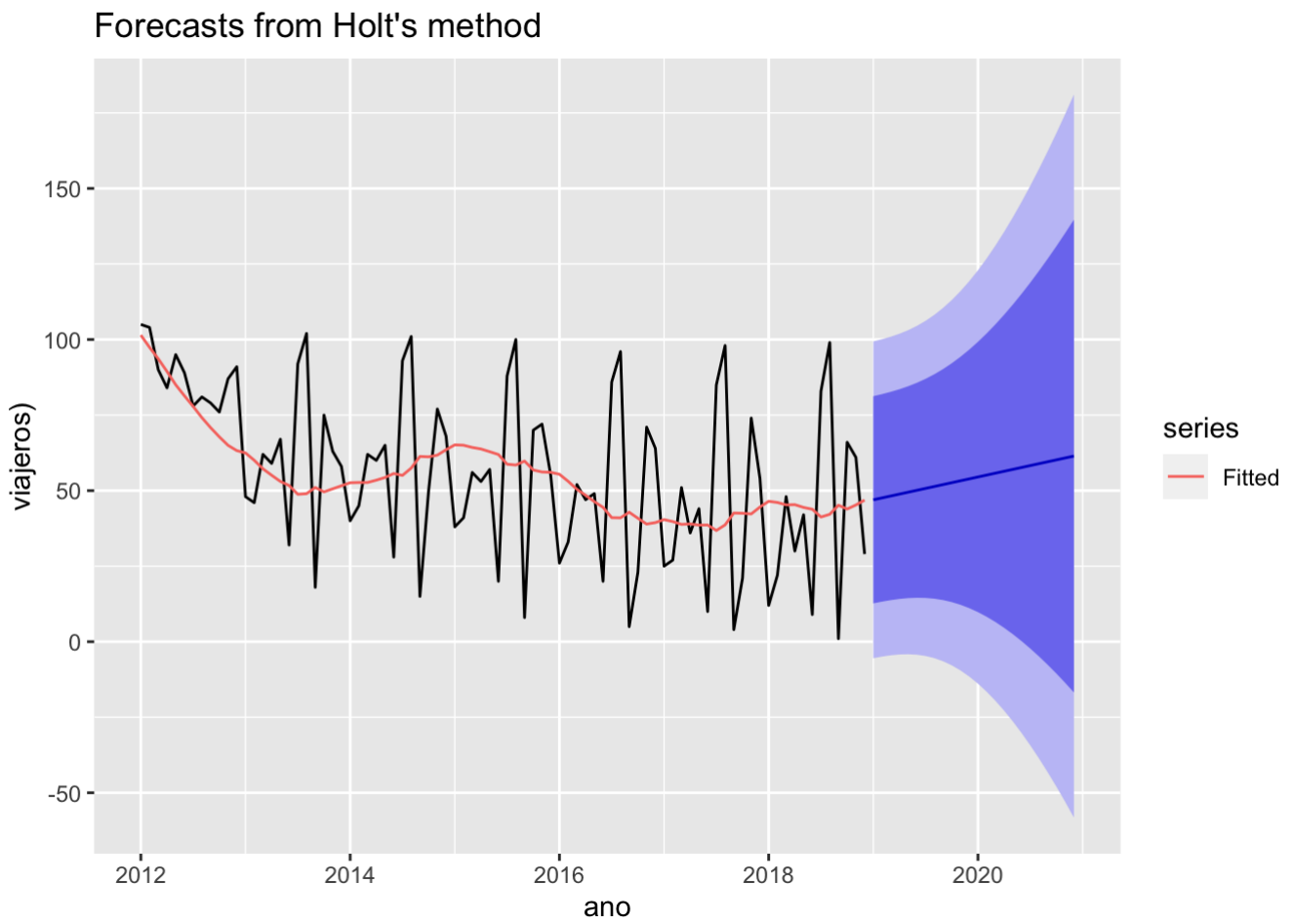



```
knitr::kable(metro_s1$model$par, digits =4,caption = "Parámetros del modelo")
```

Parámetros del modelo

	x
alpha	0.0926
l	80.1260

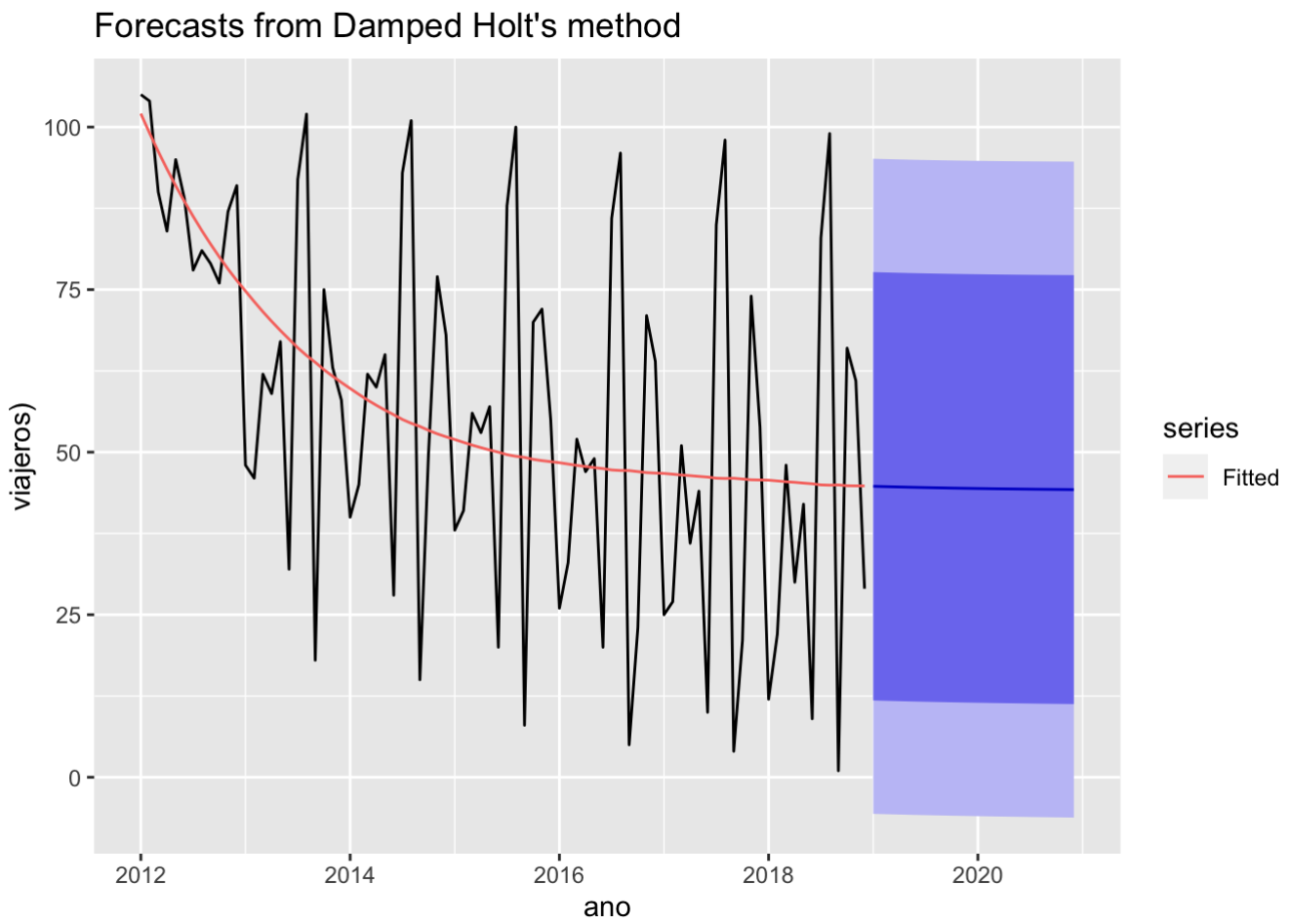
```
# 4.2)
metro_sh <- holt(train, h=24)
autoplot(metro_sh) + autolayer(fitted(metro_sh), series="Fitted") + ylab("viajeros)")
+ xlab("ano")
```




```
print(metro_sh)
```

```
##          Point Forecast      Lo 80     Hi 80      Lo 95     Hi 95
## Jan 2019       46.96314  12.692487  81.23379  -5.449300  99.37557
## Feb 2019       47.59234  13.262836  81.92184  -4.910105 100.09479
## Mar 2019       48.22154  13.759987  82.68310  -4.482860 100.92595
## Apr 2019       48.85075  14.155670  83.54582  -4.210795 101.91229
## May 2019       49.47995  14.423112  84.53679  -4.134858 103.09476
## Jun 2019       50.10915  14.537841  85.68047  -4.292474 104.51078
## Jul 2019       50.73836  14.478516  86.99820  -4.716284 106.19300
## Aug 2019       51.36756  14.227636  88.50749  -5.433052 108.16817
## Sep 2019       51.99676  13.772025  90.22150  -6.462930 110.45646
## Oct 2019       52.62597  13.103000  92.14894  -7.819195 113.07113
## Nov 2019       53.25517  12.216221  94.29412  -9.508487 116.01883
## Dec 2019       53.88437  11.111244  96.65751 -11.531484 119.30023
## Jan 2020       54.51358   9.790881  99.23628 -13.883885 122.91104
## Feb 2020       55.14278   8.260474 102.02509 -16.557520 126.84308
## Mar 2020       55.77199   6.527160 105.01681 -19.541476 131.08545
## Apr 2020       56.40119   4.599215 108.20316 -22.823094 135.62547
## May 2020       57.03039   2.485507 111.57528 -26.388811 140.44959
## Jun 2020       57.65960   0.195072 115.12412 -30.224809 145.54400
## Jul 2020       58.28880  -2.263196 118.84079 -34.317486 150.89508
## Aug 2020       58.91800  -4.880752 122.71676 -38.653773 156.48978
## Sep 2020       59.54721  -7.649529 126.74394 -43.221331 162.31574
## Oct 2020       60.17641 -10.562003 130.91482 -48.008656 168.36147
## Nov 2020       60.80561 -13.611221 135.22245 -53.005113 174.61634
## Dec 2020       61.43482 -16.790796 139.66043 -58.200933 181.07057
```

```
# 4.3)
metro_shd <- holt(train, damped=TRUE, h=24)
autoplot(metro_shd) + autolayer(fitted(metro_shd), series="Fitted") + ylab("viajero
s)") + xlab("ano")
```




```
knitr::kable(metro_shd$model$par, digits =4,caption = "parámetros Damped holt")
```

parámetros Damped holt

	x
alpha	0.0011
beta	0.0011
phi	0.9473
l	105.2415
b	-3.3457

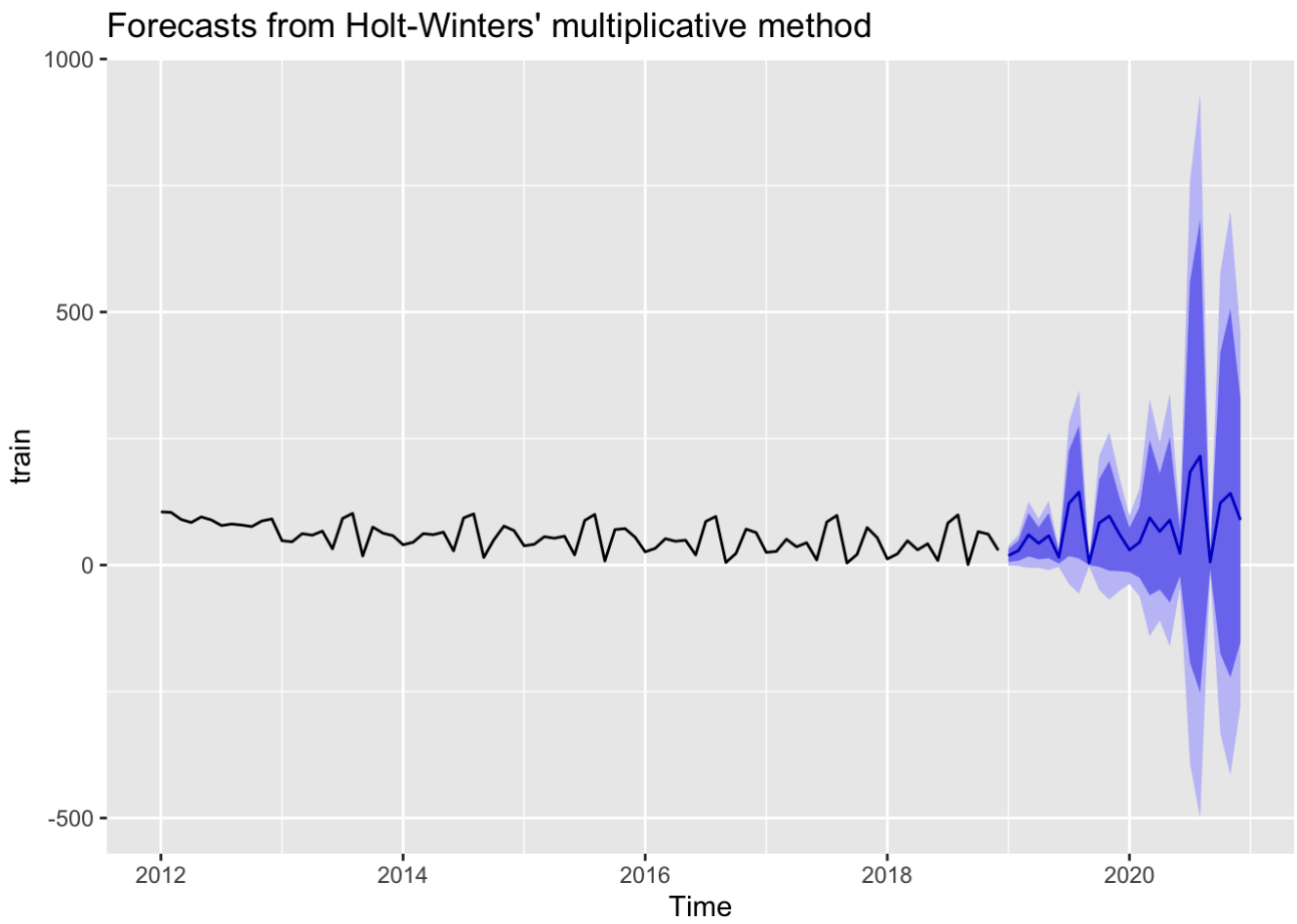
```
knitr::kable(accuracy(metro_shd), digits =4,caption = "Medidas de bon dad del ajuste
 damped-holt")
```

Medidas de bon dad del ajuste damped-holt

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-0.5988	24.9249	19.7495	-107.1839	130.3631	1.6948	-0.133

```
# 4.4)
metro_shw <- hw(train, h=24, seasonal="multiplicative",level = c(80, 95))
autoplot(metro_shw)
```




```
knitr::kable(metro_shw$model$par, digits =4,caption = "parámetros Holt-winters")
```

parámetros Holt-winters

	x
alpha	0.0797
beta	0.0586
gamma	0.5426
l	148.7985
b	-3.1376
s0	1.3155
s1	1.2690
s2	1.1718

	x
s3	0.3311
s4	1.3579
s5	1.1261
s6	0.9216
s7	0.9720
s8	0.9000
s9	0.8565
s10	0.8878

```
knitr::kable(accuracy(metro_shw), digits =4,caption = "Medidas Holt-winters")
```

Medidas Holt-winters

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.8053	17.0152	12.9369	-13.7378	33.5516	1.1102	-0.0173

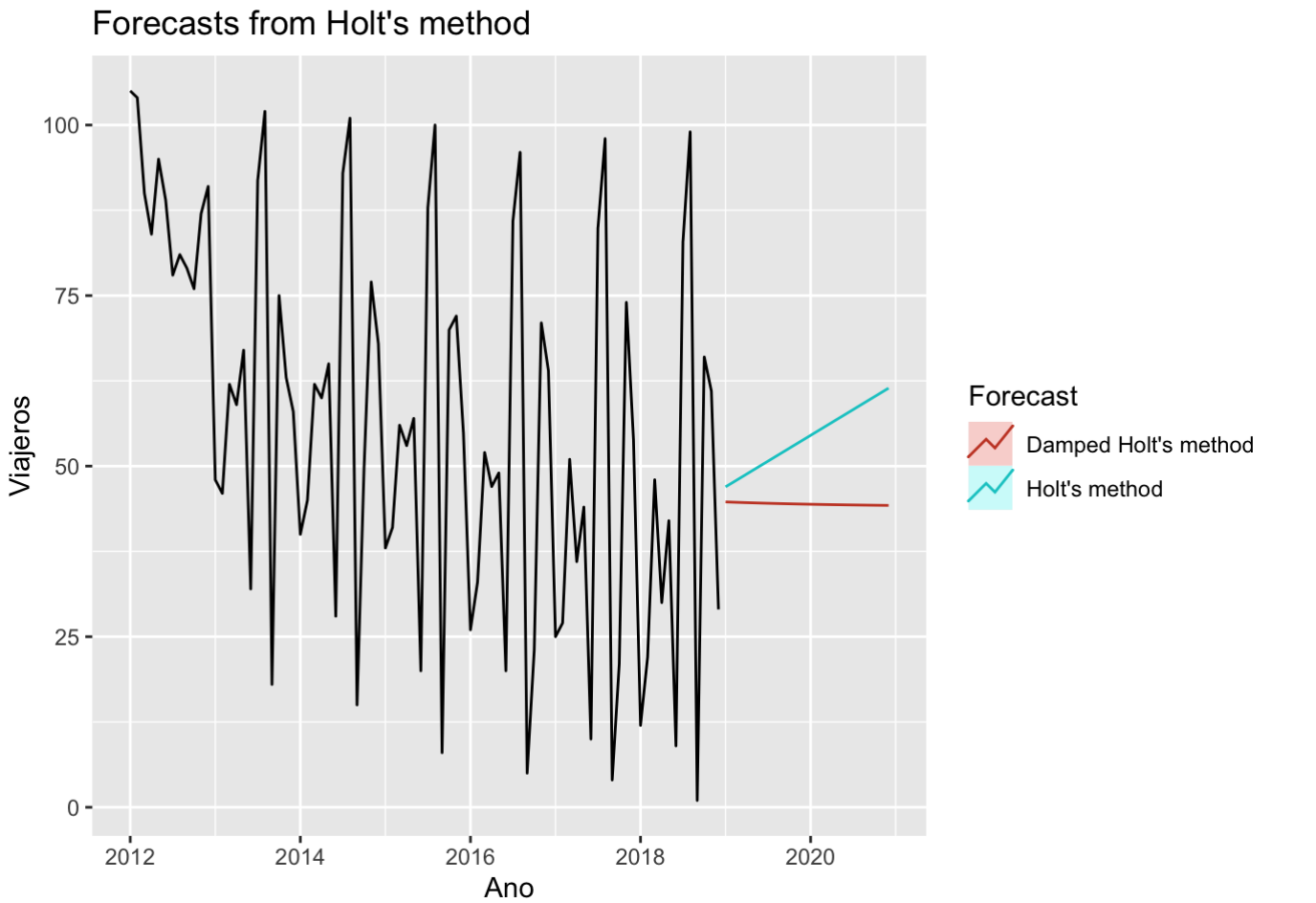
```
knitr::kable(metro_shw, digits =4,caption = "Predicciones ")
```

Predicciones

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan 2019	18.6932	5.7483	31.6381	-1.1043	38.4907
Feb 2019	28.5235	8.5467	48.5003	-2.0283	59.0753
Mar 2019	59.8028	17.0439	102.5616	-5.5913	125.1968
Apr 2019	42.9015	11.2615	74.5416	-5.4877	91.2908
May 2019	58.1093	13.4580	102.7605	-10.1789	126.3975
Jun 2019	15.1627	2.9151	27.4103	-3.5684	33.8938
Jul 2019	121.9577	17.6547	226.2608	-37.5600	281.4755
Aug 2019	144.1350	12.9147	275.3553	-56.5491	344.8192
Sep 2019	4.1268	0.1129	8.1408	-2.0120	10.2656
Oct 2019	83.1829	-3.4466	169.8124	-49.3055	215.6714
Nov 2019	96.9349	-11.2683	205.1381	-68.5476	262.4174
Dec 2019	61.2185	-12.0399	134.4770	-50.8206	173.2577
Jan 2020	29.8349	-14.1049	73.7747	-37.3652	97.0350
Feb 2020	45.0147	-24.8683	114.8976	-61.8621	151.8914
Mar 2020	93.3879	-59.6705	246.4463	-140.6948	327.4706

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Apr 2020	66.3344	-48.5630	181.2318	-109.3860	242.0548
May 2020	89.0147	-74.0363	252.0657	-160.3503	338.3797
Jun 2020	23.0237	-21.5914	67.6388	-45.2092	91.2566
Jul 2020	183.6547	-192.8826	560.1921	-392.2095	759.5189
Aug 2020	215.3535	-251.7794	682.4863	-499.0646	929.7715
Sep 2020	6.1202	-7.9231	20.1636	-15.3573	27.5977
Oct 2020	122.4954	-174.7666	419.7575	-332.1277	577.1186
Nov 2020	141.7926	-222.0127	505.5979	-414.5995	698.1847
Dec 2020	88.9787	-152.3272	330.2845	-280.0667	458.0241

```
# 4.5)
autoplot(train) + autolayer(metro_sh, series="Holt's method", PI=FALSE) + autolayer(m
etro_shd, series="Damped Holt's method", PI=FALSE) + ggtitle("Forecasts from Holt's m
ethod") + xlab("Ano") + ylab("Viajeros") + guides(colour=guide_legend(title="Forecas
t"))
```



Observemos con el método suavizado exponencial simple que la predicción permanece constante para todos los años. Vemos también que los valores ajustados y las predicciones con este método no son muy buenos.

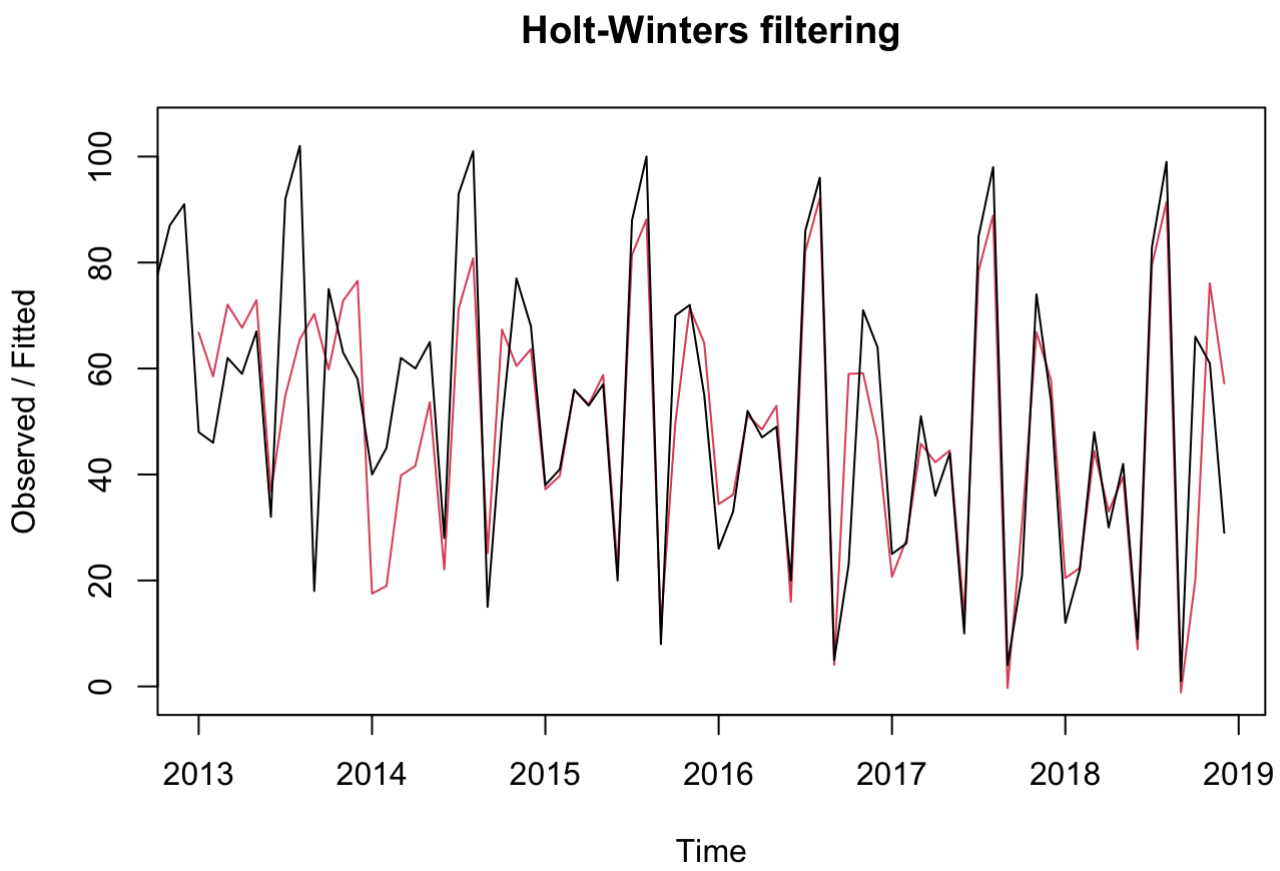
Por otra parte, vemos que el método de la función holt da unas predicciones más adecuadas, puesto que tiene en cuenta los cambios en la tendencia.

Pero si queremos predecir el número de viajeros en el metro de Sevilla un año después del último observado, utilizaremos el modelo de Holt-winters multiplicativo porque la serie es estacional.

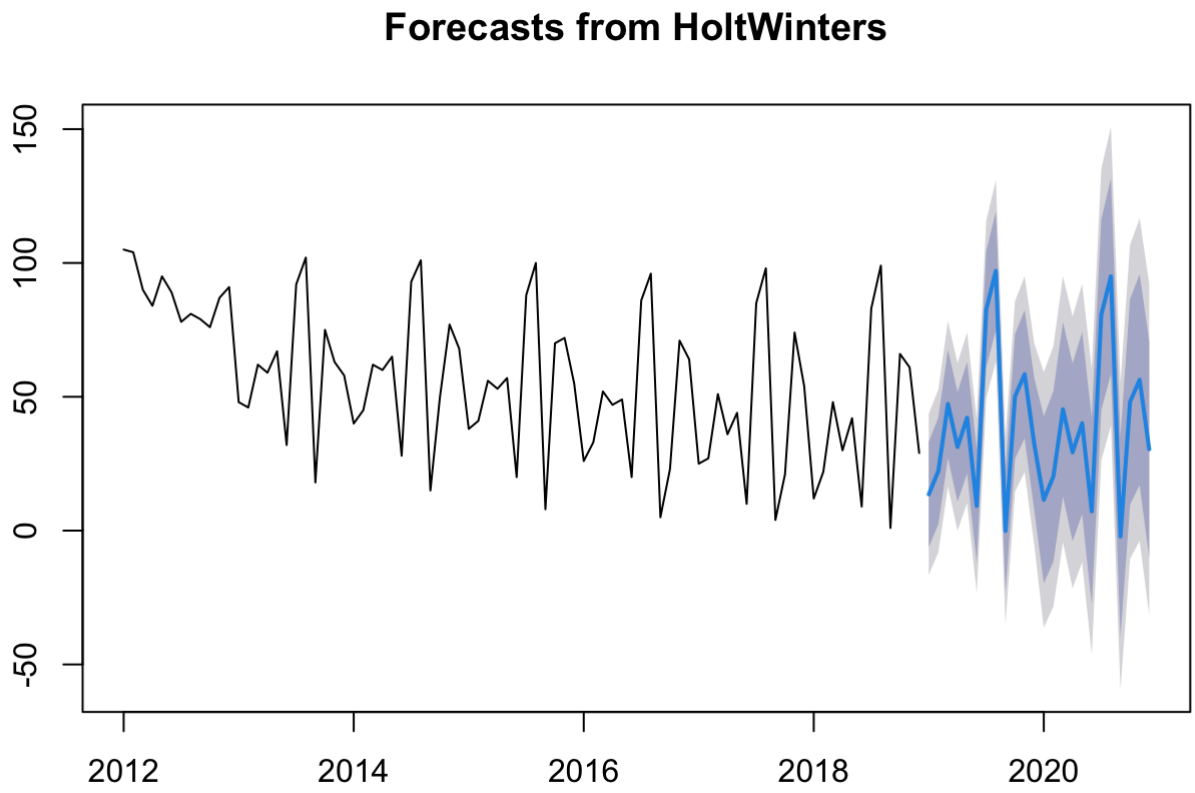
Como podemos observar en las gráficas, este método da unas predicciones más adecuadas puesto que tiene en cuenta los cambios en la tendencia. Así que comprobamos en la representación gráfica final que el modelo de suavizado exponencial más adecuado es el Holt-Winters. De hecho, es fqvorqble a la hora de predecir los picos estacionales del periodo elegido.

Comprobamos nuestra observación con otra representación gráfica a continuación:

```
# 4.6)
LESB=HoltWinters(train)
plot(LESB)
```



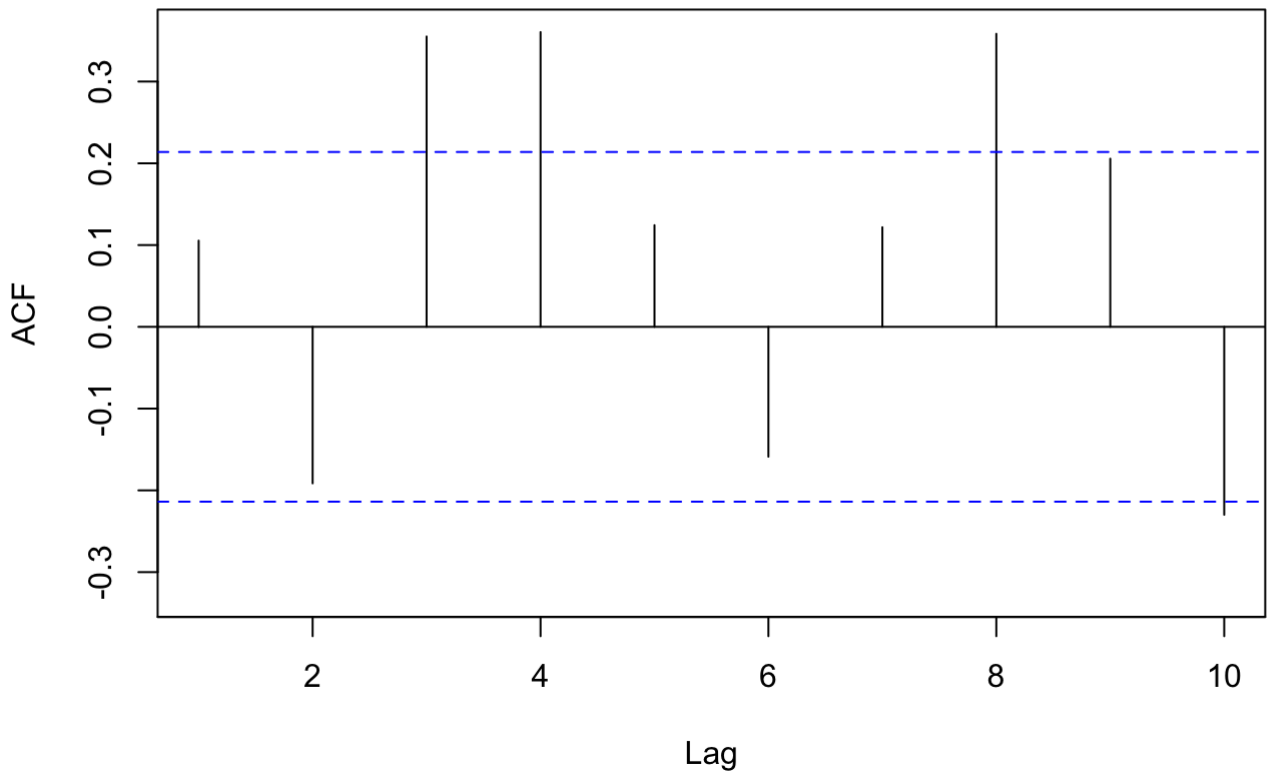

```
# 4.6)
metroforecast= forecast(LESB,h=24)
plot(metroforecast)
```



La representación confirma nuestra observación, el modelo de suavizado exponencial más adecuado es el Holt-Winters.

5. Representar la serie y los correlogramas. Decidir que modelo puede ser ajustado. Ajustar el modelo adecuado comprobando que sus residuales están incorrelados. (Sintaxis, tablas de los parámetros estimados y gráficos).

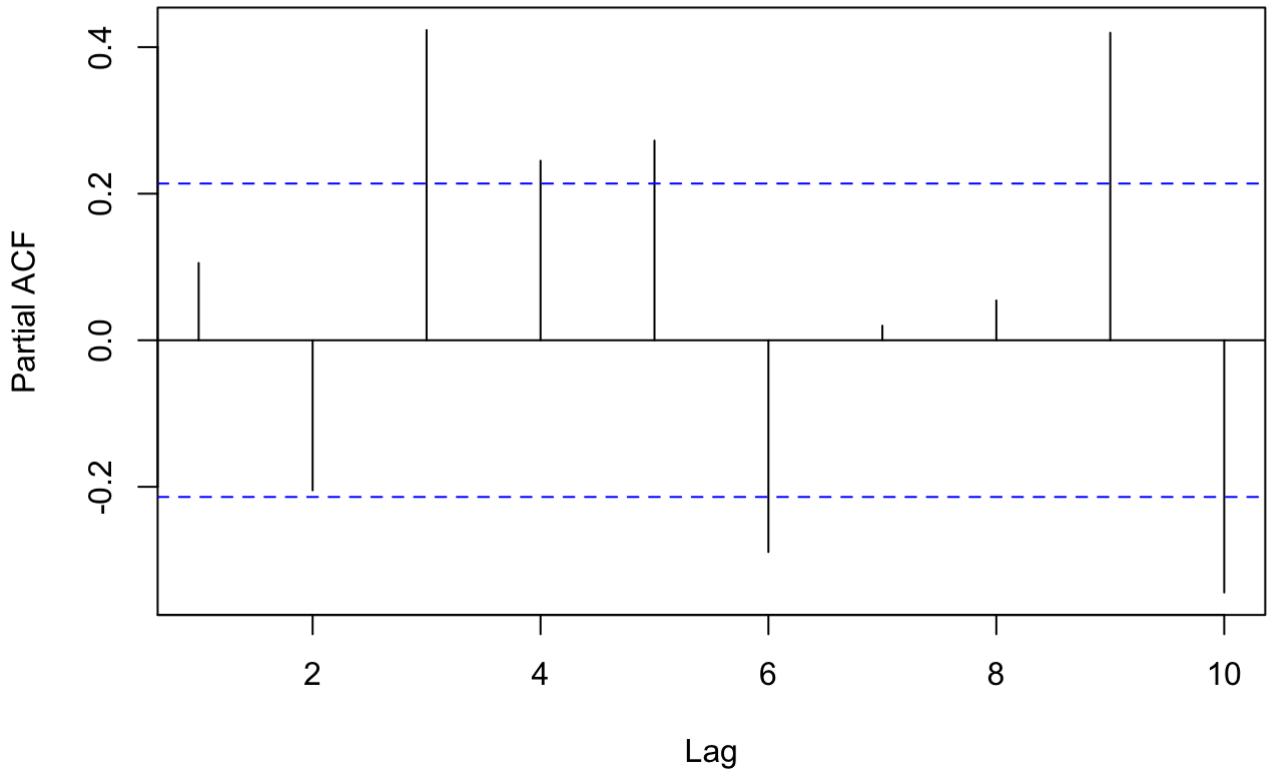
```
# 5.1)
corr<-Acf(train, lag=10)
```




```
print(corr)
```

```
##
## Autocorrelations of series 'train', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000  0.105 -0.191  0.355  0.360  0.124 -0.159  0.122  0.358  0.206 -0.230
```

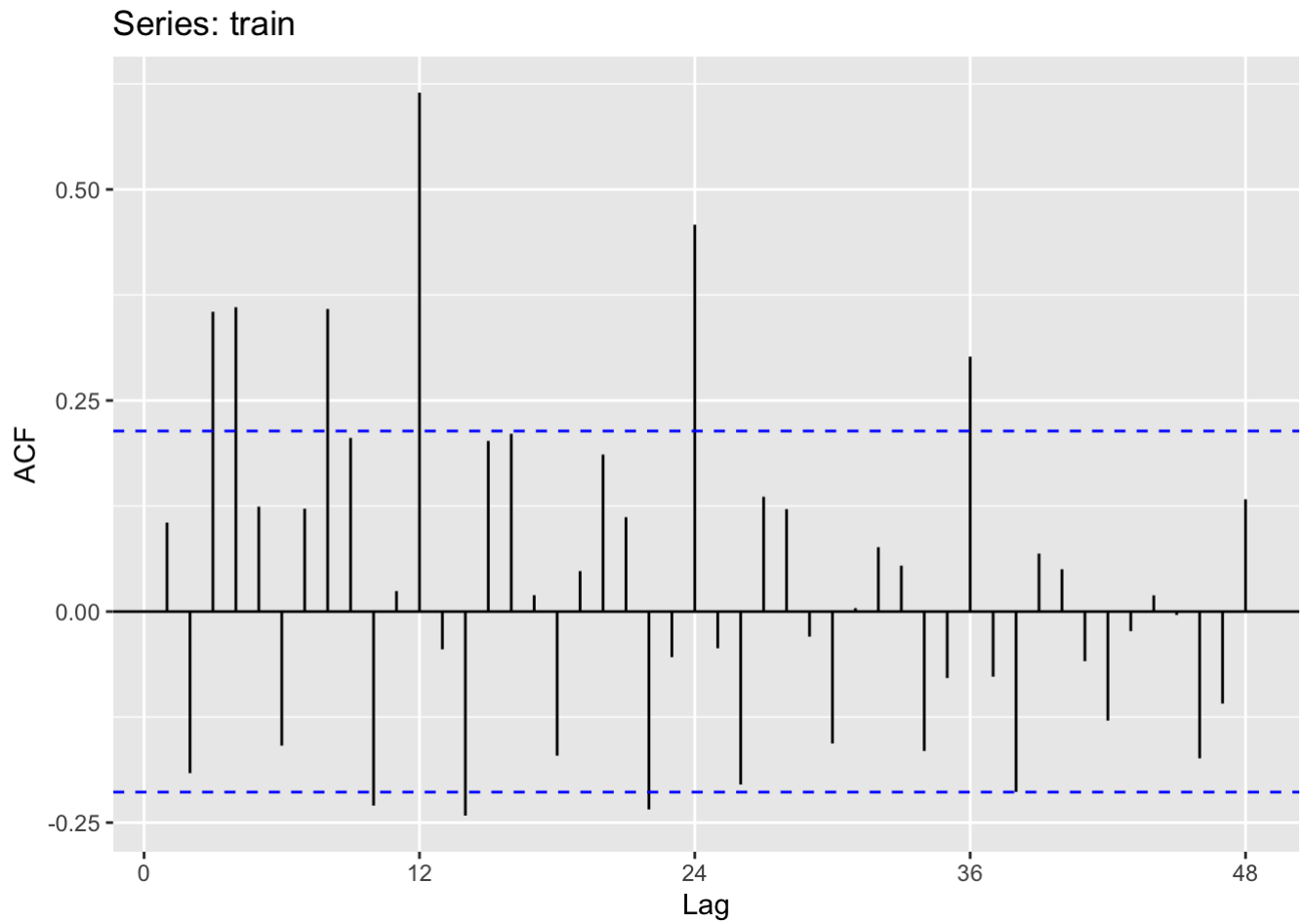
```
corrp<-Pacf(train, lag=10)
```



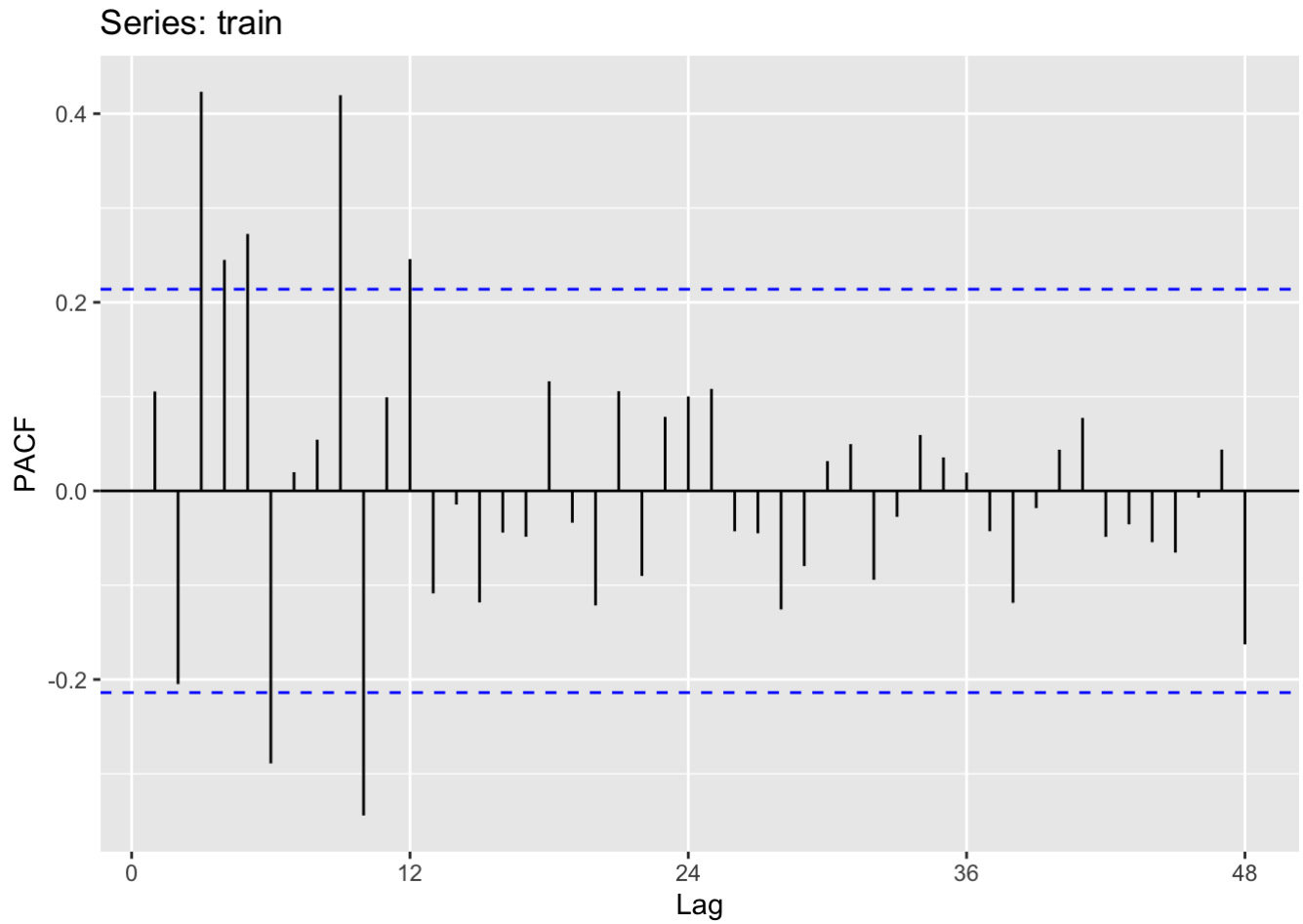

```
print(corrp)
```

```
##
## Partial autocorrelations of series 'train', by lag
##
##      1      2      3      4      5      6      7      8      9     10
##  0.105 -0.205  0.423  0.245  0.272 -0.289  0.020  0.054  0.420 -0.344
```

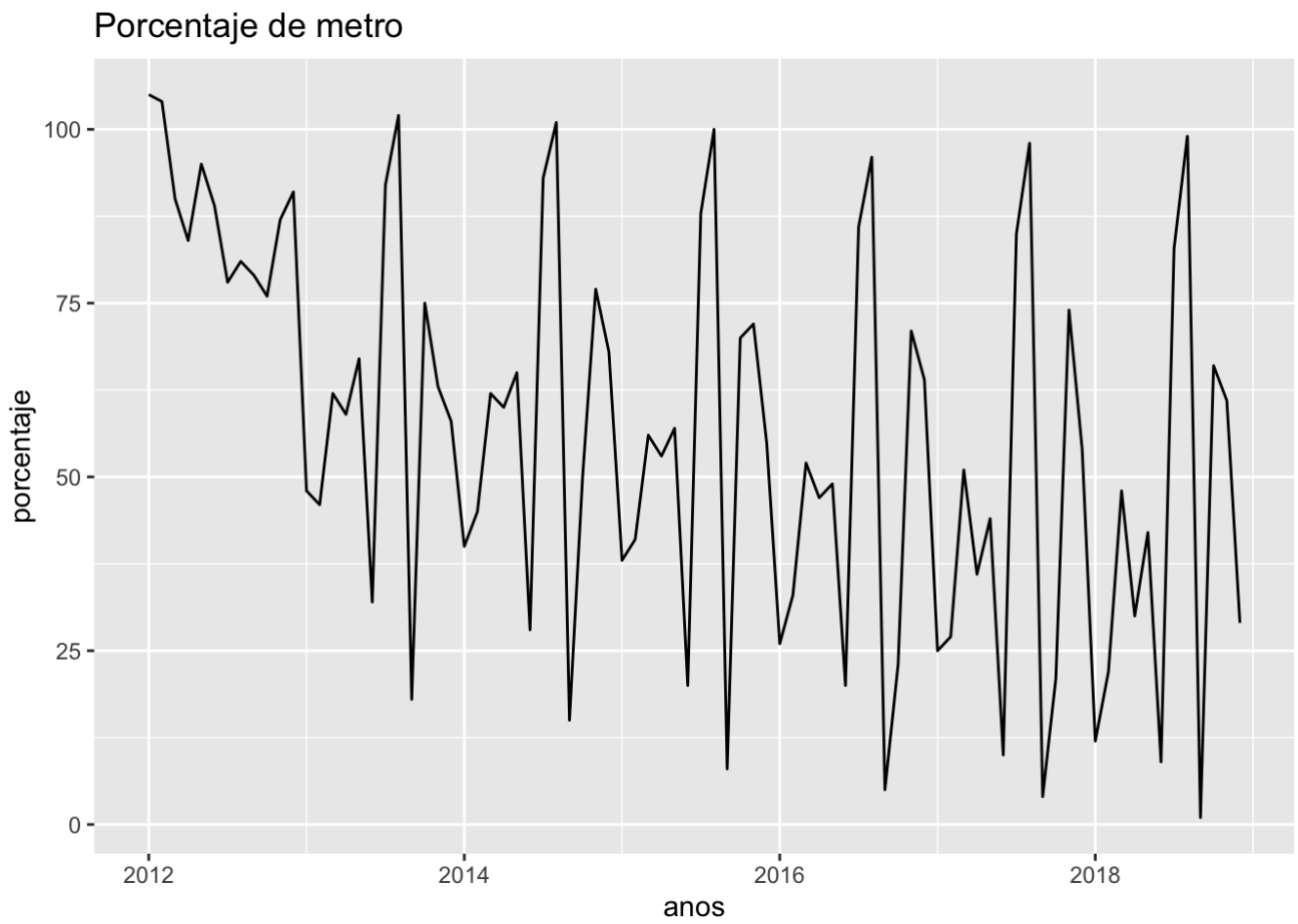
```
# 5.1.1)
ggAcf(train, lag=48)
```



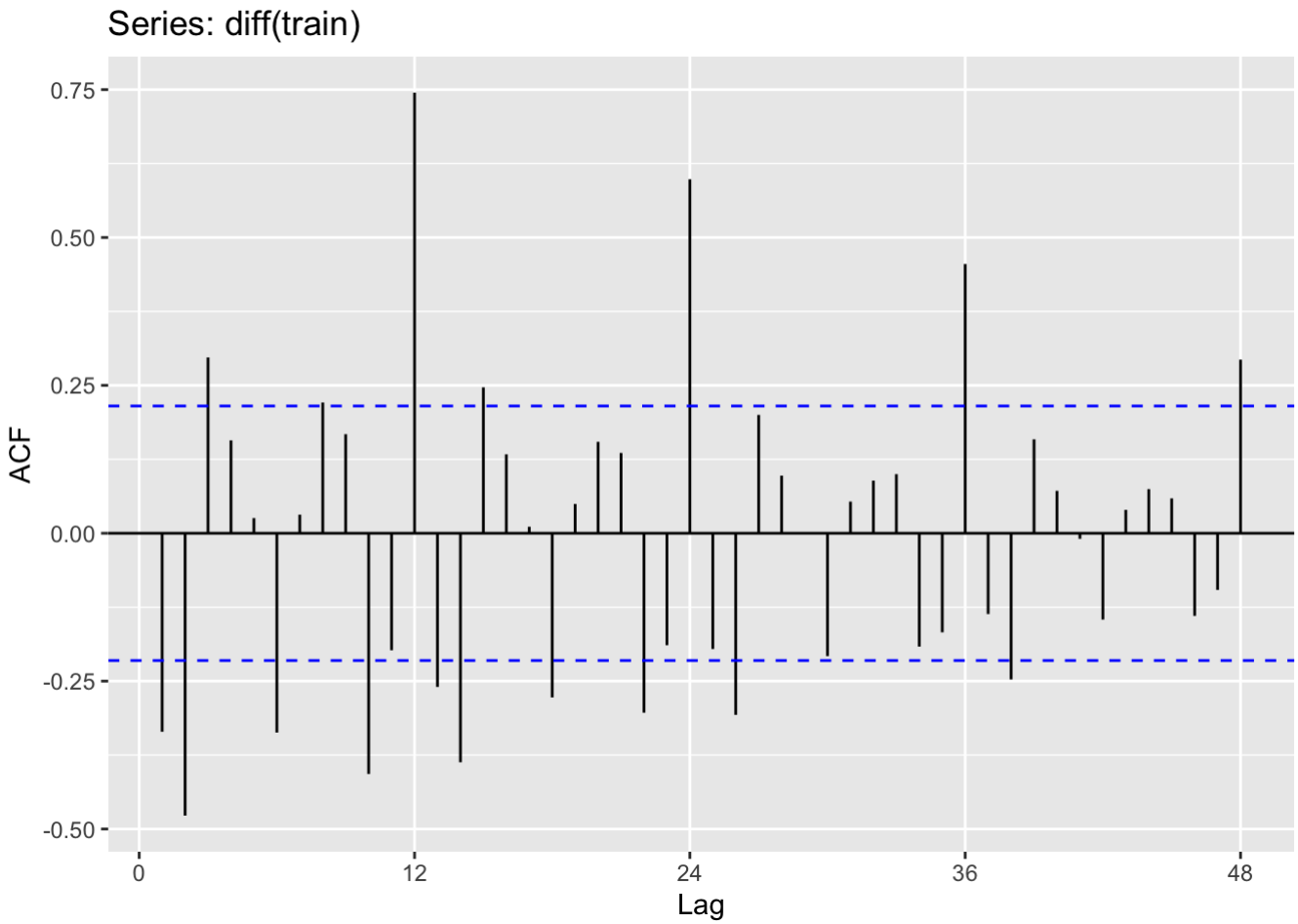

```
# 5.1.2)
ggPacf(train, lag=48)
```



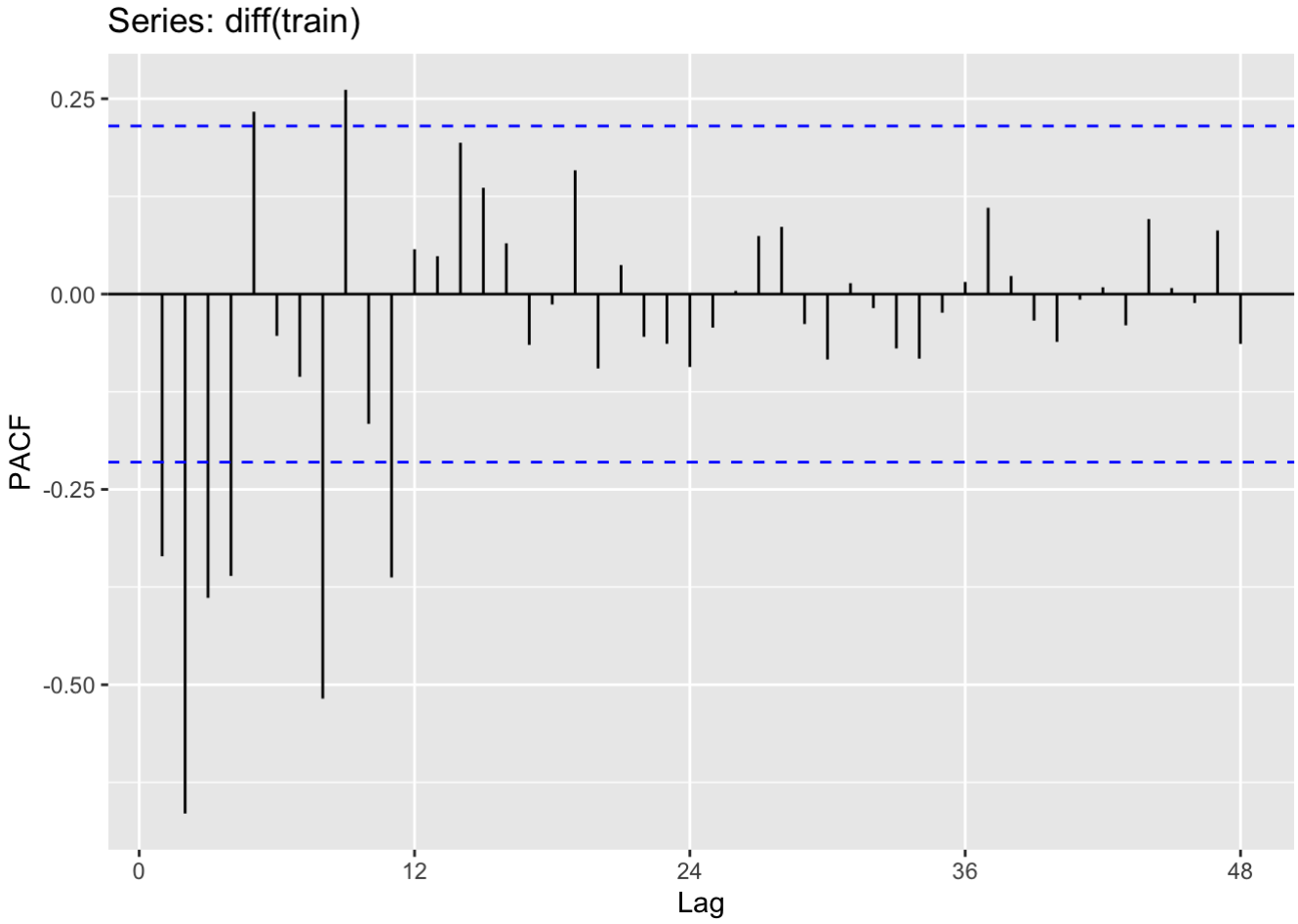
```
# 5.1.3)
autoplot(train)+ ggtitle("Porcentaje de metro") + xlab("anos") + ylab("porcentaje")
```



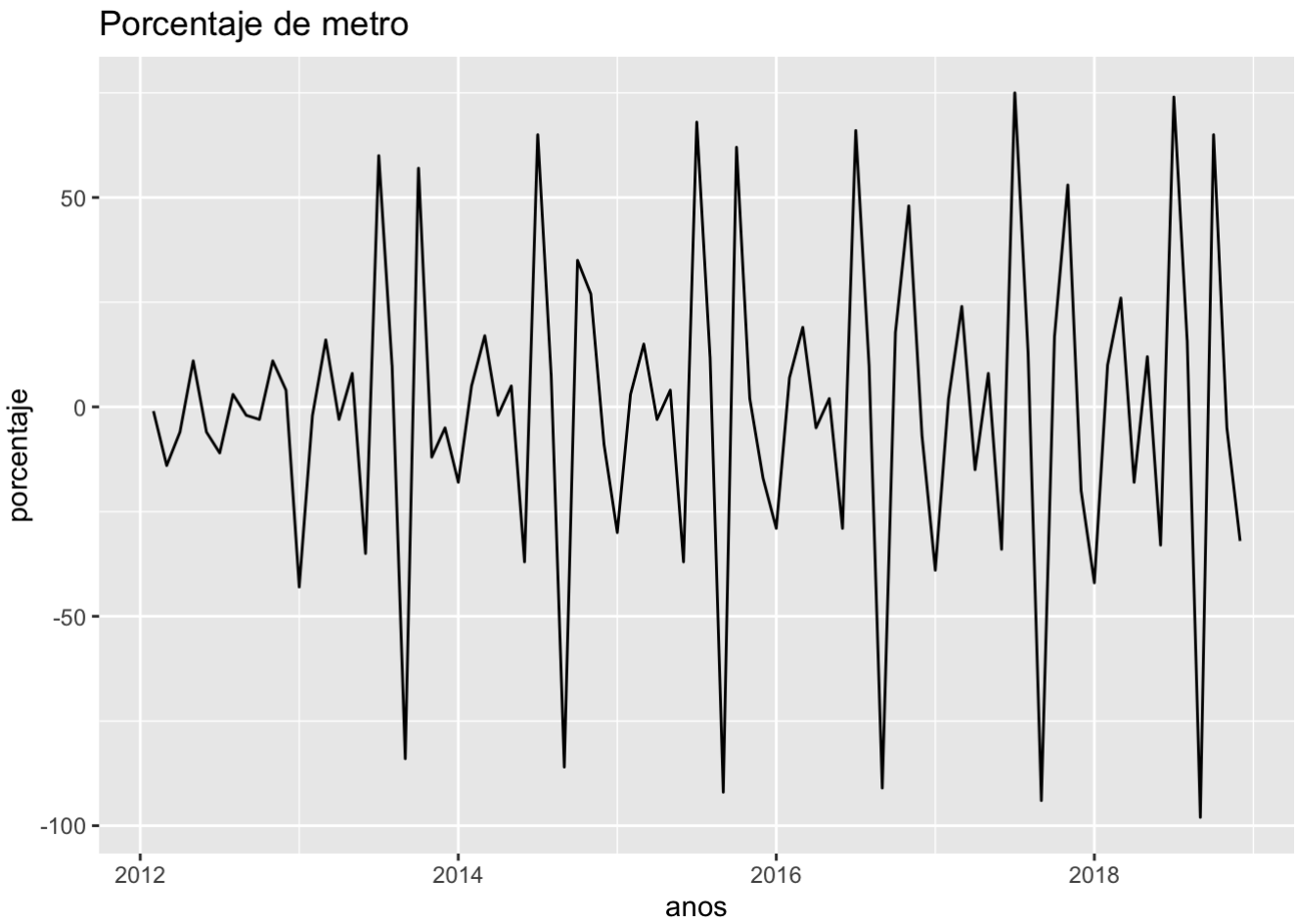

```
# 5.2)
# 5.2.1)
ggAcf(diff(train), lag=48)
```



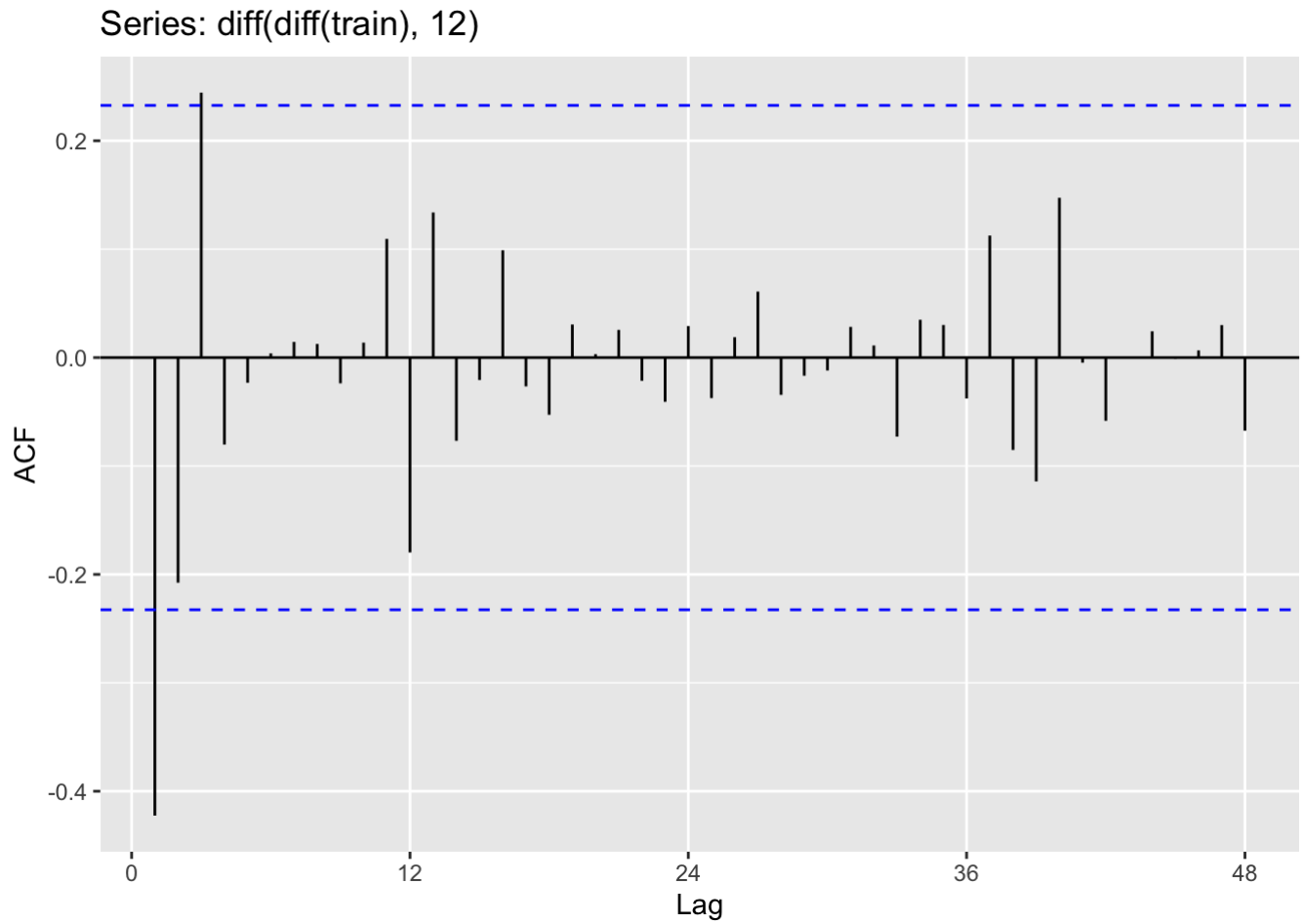

```
# 5.2.2)
ggPacf(diff(train), lag=48)
```



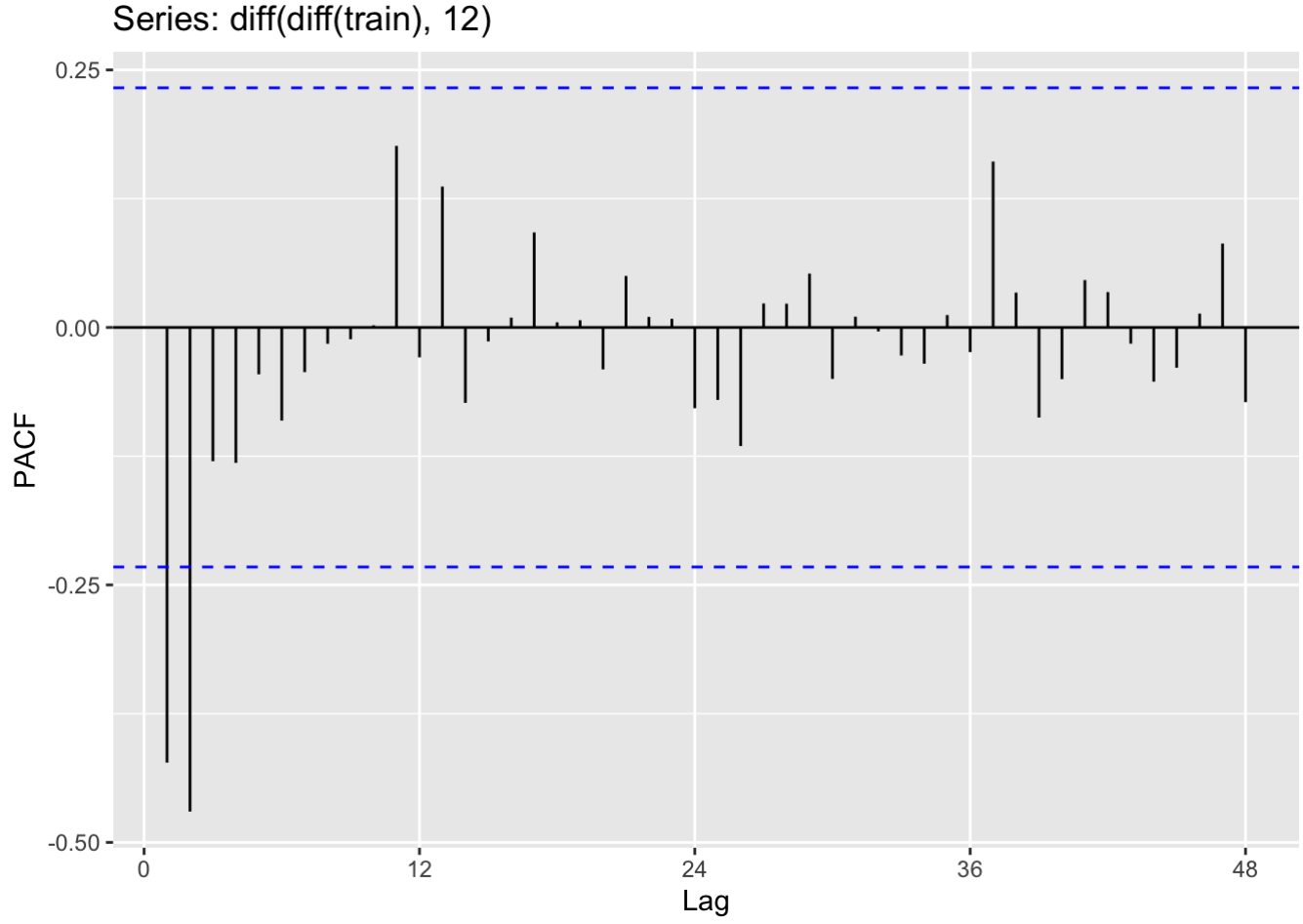
```
# 5.3.3
autoplot(diff(train))+ ggtitle("Porcentaje de metro") + xlab("anos") + ylab("porcenta
je")
```



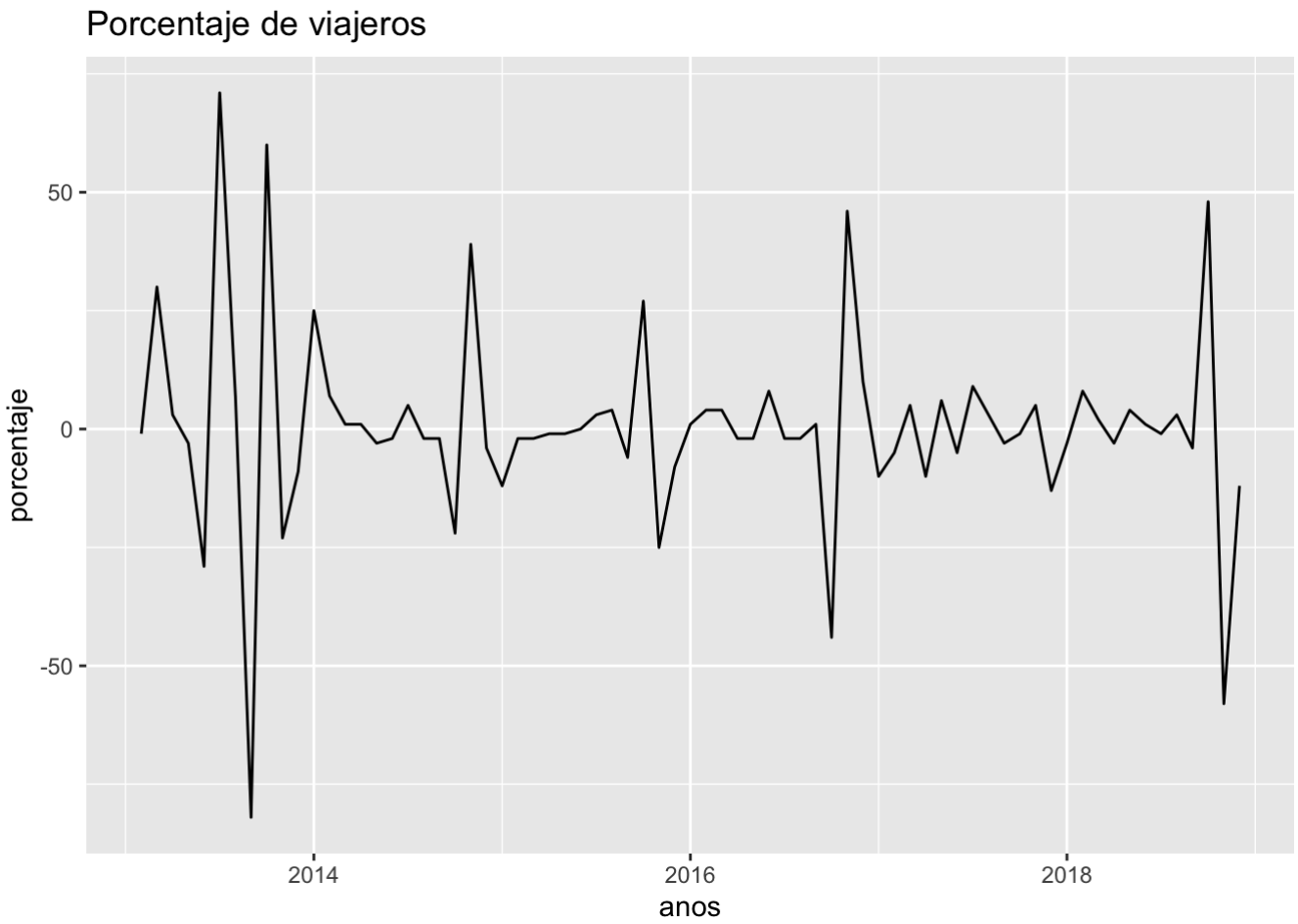

```
# 5.3)
# 5.3.1)
ggAcf(diff(diff(train),12), lag=48)
```




```
# 5.3.2)
ggPacf(diff(diff(train),12), lag=48)
```



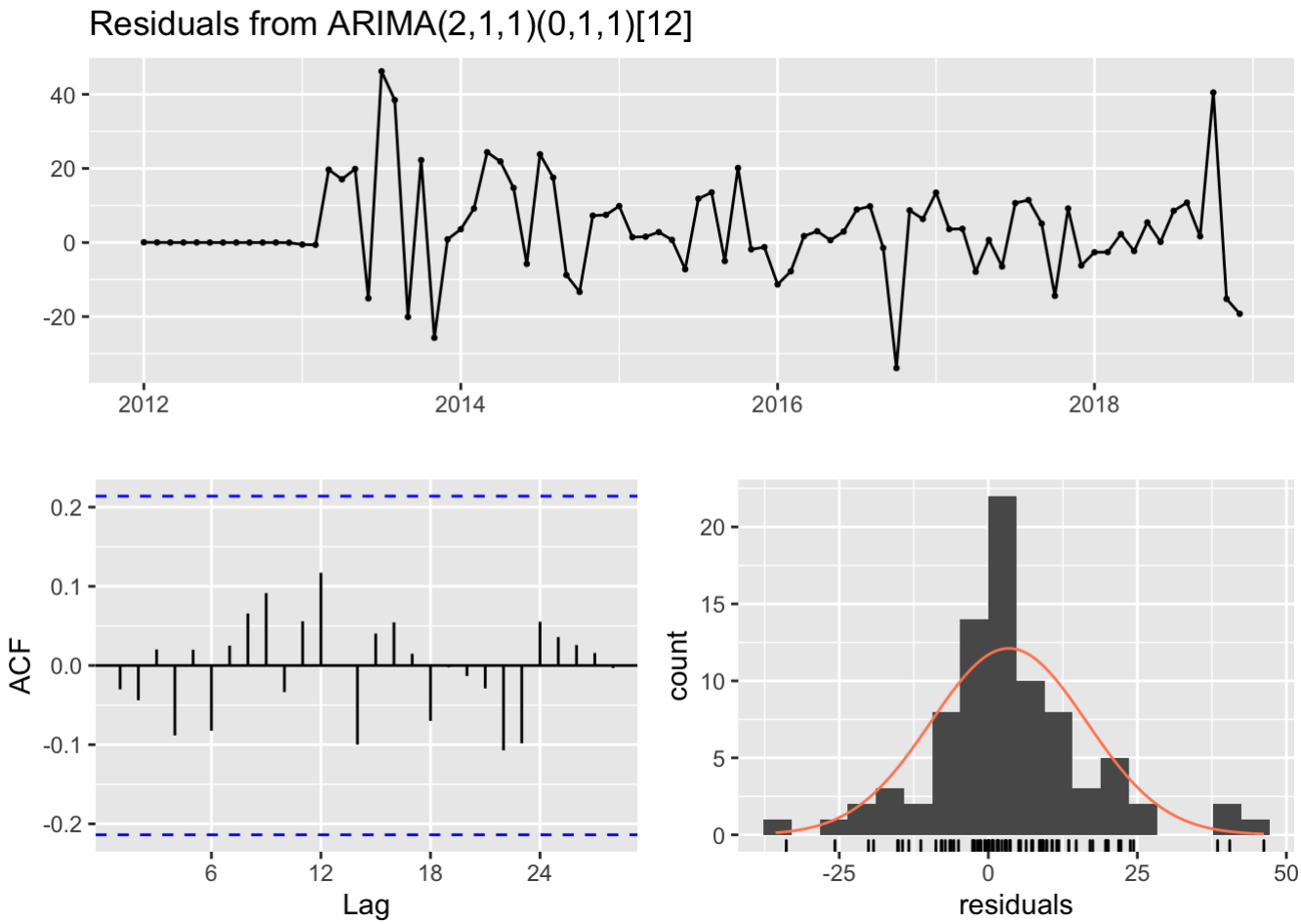
```
# 5.3.3)
autoplot(diff(diff(train),12))+ ggtitle("Porcentaje de viajeros") +
xlab("anos") + ylab("porcentaje")
```



- En el autocorrelograma simple, se observa un comportamiento repetitivo de las autocorrelaciones cada 12 meses. Puesto que en el gráfico de la serie hemos visto que la media no es constante porque la serie tiene tendencia y el ACF decrece de forma lenta, es necesario hacer una diferenciación Además, el autoplot muestra una clara no estacionariedad.
- En la serie diferenciada, mediante una diferenciación de orden estacional, podemos ver que el comportamiento es parecido al autocorrelograma simple.
- Con la serie doblemente diferenciada vemos que el proceso ya es estacionario. Observamos que las autocorrelaciones decrecen de forma más rápida: el ACF se corta después del 1 y el PACF corta después del 2. Por ello, nuestro candidato a ajustar sera: ARIMA(1,1,0)(0,1,1)12.

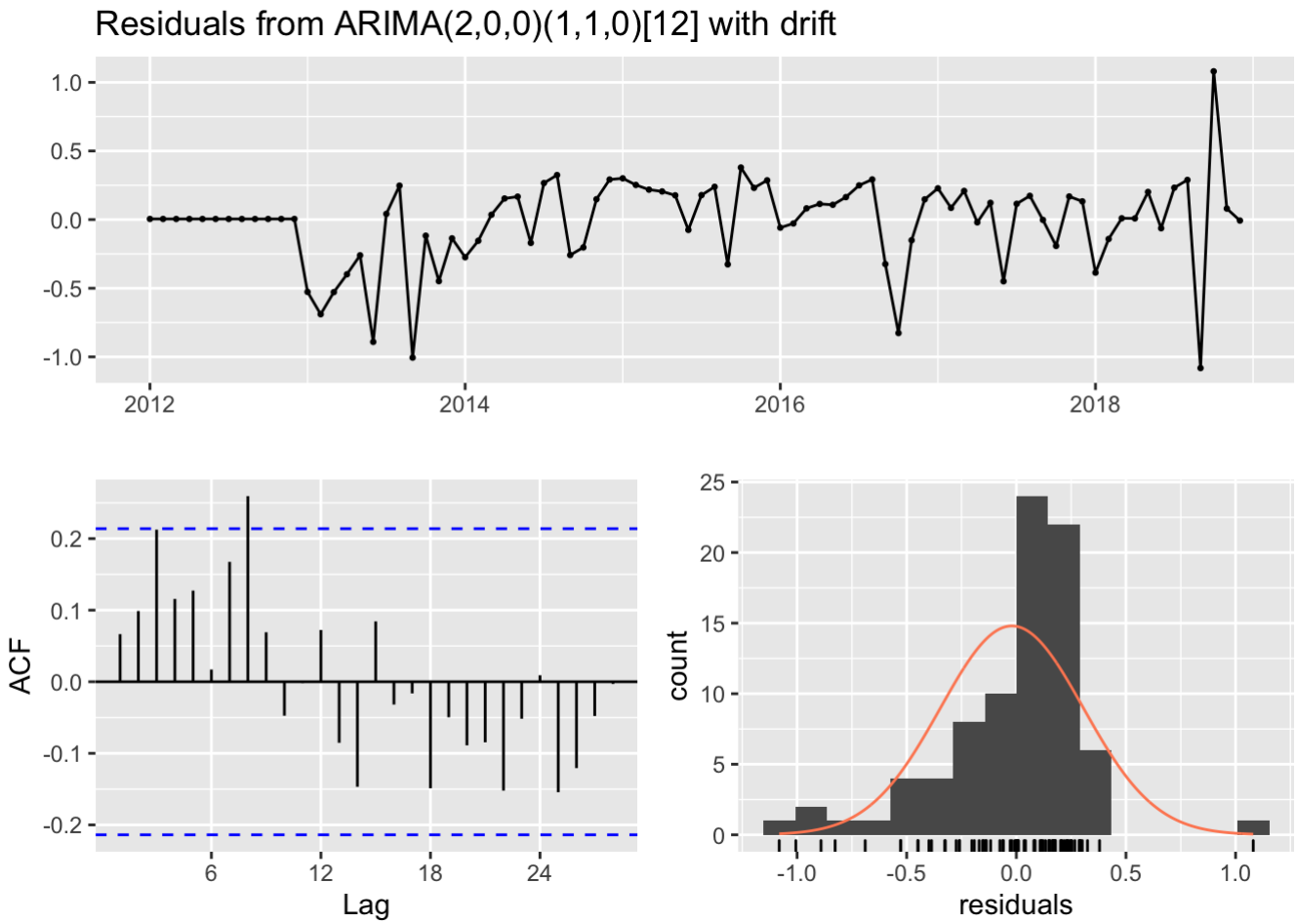
Vamos a usar la función auto.arima. Esta función busca a través de combinaciones de parámetros de pedido y selecciona el conjunto que optimiza los criterios de ajuste del modelo. De hecho, la función auto.arima encuentra el mejor modelo Arima ajustando todos los órdenes hasta que consigue que los residuos están incorrelados.

```
# 5.4)
# Model 1
fit <- auto.arima(train)
checkresiduals(fit)
```



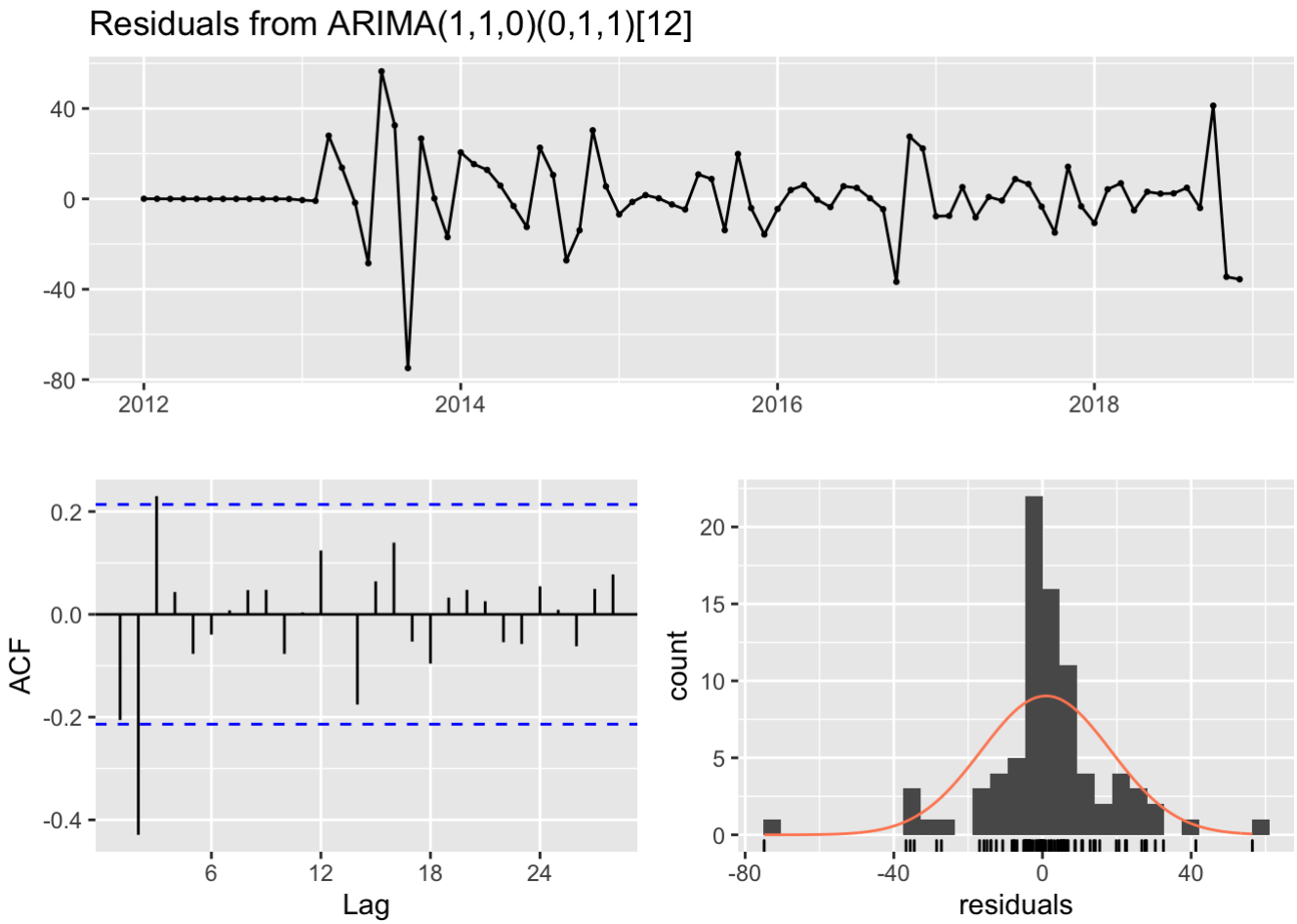

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,1)(0,1,1)[12]
## Q* = 6.2625, df = 13, p-value = 0.9361
##
## Model df: 4.   Total lags used: 17
```

```
# Model 1 (log)
fit1 <- auto.arima(log(train))
checkresiduals(fit1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0)(1,1,0)[12] with drift
## Q* = 22.057, df = 13, p-value = 0.05448
##
## Model df: 4.   Total lags used: 17
```

```
# Model 2 - Ajuste del ARIMA(1,1,0)
fit2 <- Arima((train),c(1,1,0),seasonal=c(0,1,1))
checkresiduals(fit2)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(0,1,1)[12]
## Q* = 34.015, df = 15, p-value = 0.003389
##
## Model df: 2.   Total lags used: 17
```

```
# 5.5)
knitr::kable(accuracy(fit), digits =4,caption = "Medidas de ajuste")
```

Medidas de ajuste

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	3.4371	13.41	9.0729	1.4172	24.1369	0.7786	-0.0302

Puesto que el contraste de Ljung-Box nos da un p-valor >0.05 en el Modelo 1, aceptamos la hipótesis de que los residuos están incorrelados, lo que también podemos ver en su autocorrelograma.

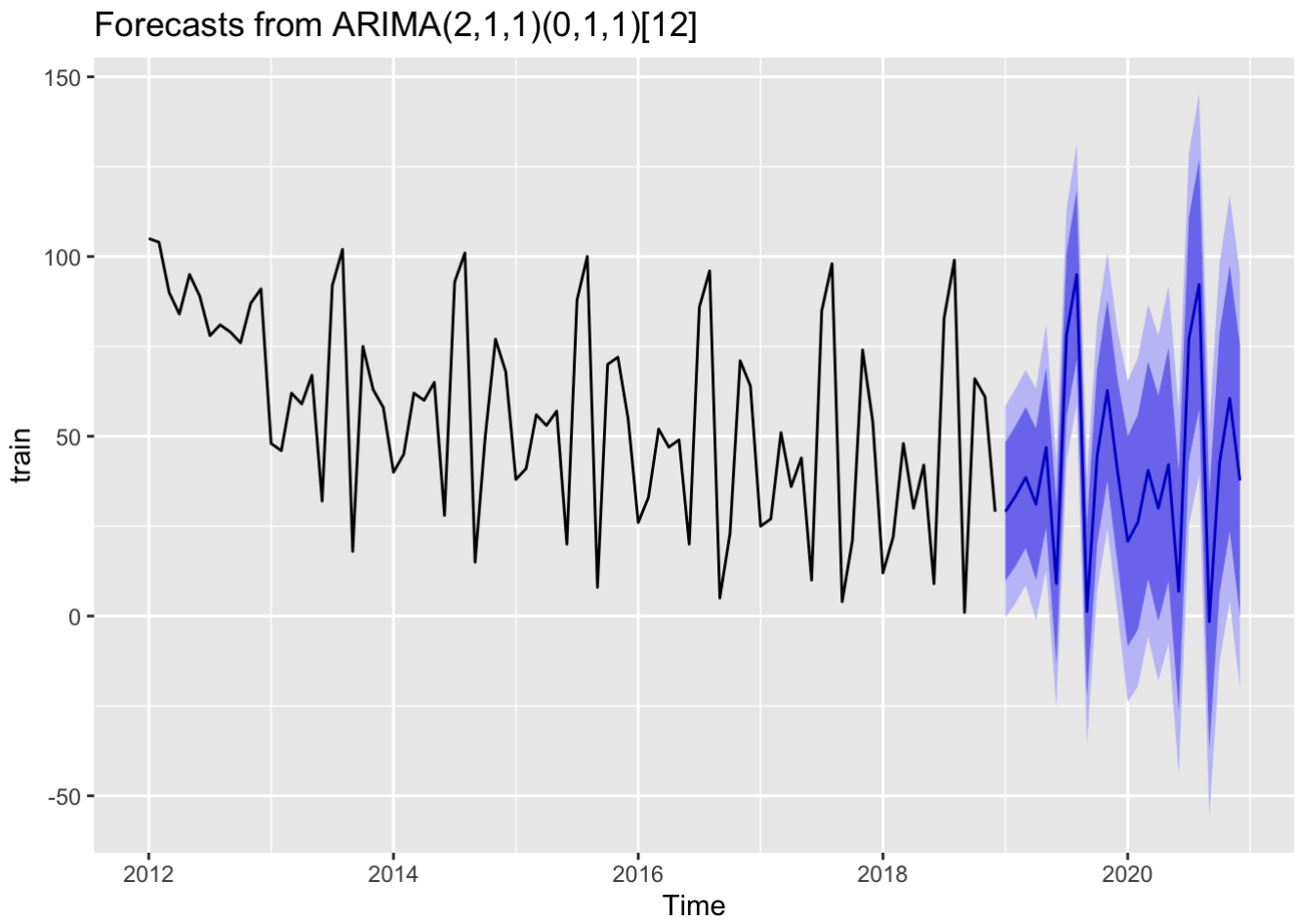
Por otro lado, el modelo 2 (fit2) da una autocorrelación, ya que que el pvalor es mucho menor de 0.05 rechazamos que los residuos están incorrelados, lo que implica que el modelo no explica toda la dependencia de la serie. Rechazamos el modelo 2.

Finalmente, los residuos son independientes y que el modelo 1 (fit) y el modelo 1 (fit1) log respetan la hipótesis. Elegimos el modelo 1, lo confirma la autocorrelation test box pierce, ya que el valor p es mayor que 0.05 (0.9361).

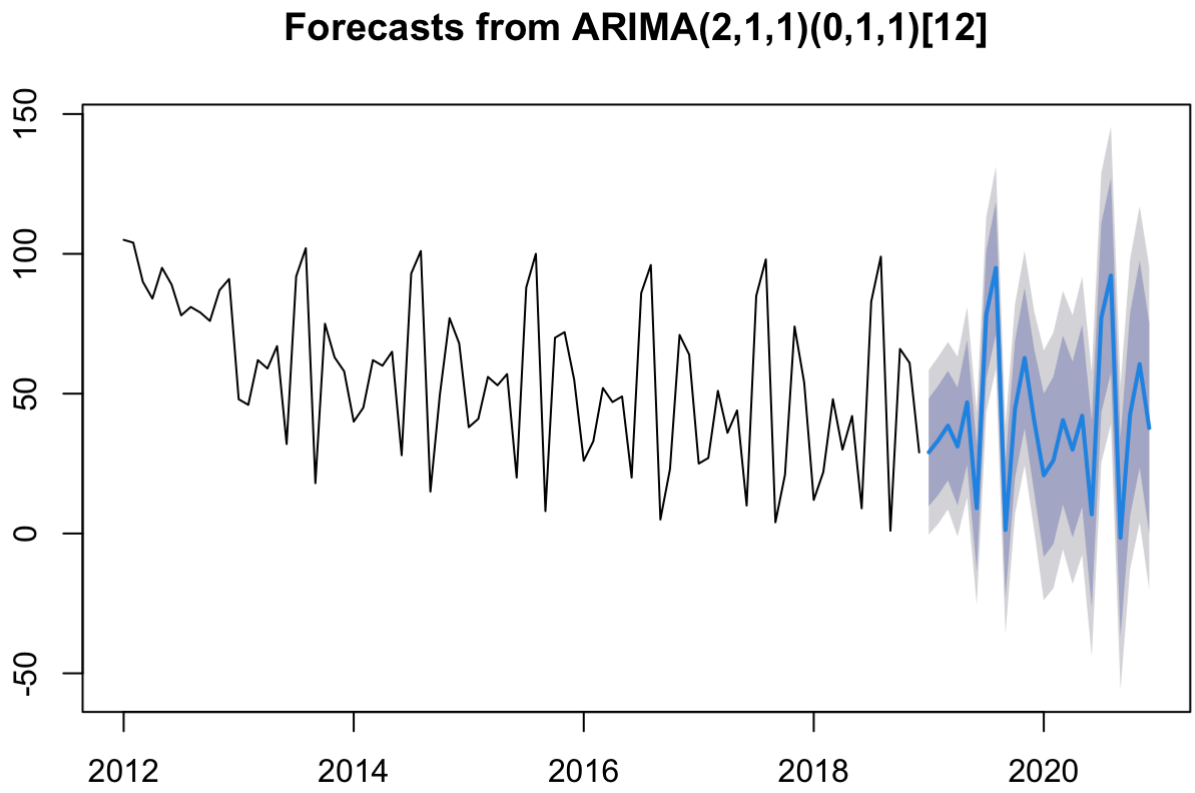
En conclusión, el modelo 1 (fit) puede valer.

6. Calcular las predicciones y los intervalos de confianza para las unidades de tiempo que se considere oportuno, dependiendo de la serie, siguientes al último valor observado. Representarlas gráficamente.

```
# 6.1)
autoplot(forecast(fit,h=24))
```




```
# 6.2)
metropred=forecast(fit,h = 24)
plot(metropred)
```




```
# 6.3)
predi<-forecast(fit,h=24)
knitr::kable(predi, digits =4,caption = "Predicciones ")
```

Predicciones

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Jan 2019	29.0287	9.7846	48.2727	-0.4026	58.4599
Feb 2019	33.4013	13.9024	52.9002	3.5803	63.2224
Mar 2019	38.5563	18.9876	58.1249	8.6286	68.4839
Apr 2019	31.1173	10.1204	52.1141	-0.9946	63.2292
May 2019	46.8904	24.6622	69.1186	12.8953	80.8855
Jun 2019	9.0586	-13.3517	31.4688	-25.2149	43.3321
Jul 2019	78.2960	55.4801	101.1119	43.4020	113.1899
Aug 2019	94.9499	71.3460	118.5538	58.8508	131.0490
Sep 2019	1.2479	-22.8470	25.3428	-35.6020	38.0978
Oct 2019	44.5642	20.0847	69.0437	7.1260	82.0024
Nov 2019	62.7471	37.7496	87.7447	24.5167	100.9776
Dec 2019	40.2380	14.7249	65.7511	1.2191	79.2569
Jan 2020	20.8084	-8.3835	50.0003	-23.8368	65.4536

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Feb 2020	26.1285	-3.7484	56.0054	-19.5643	71.8213
Mar 2020	40.5397	10.3742	70.7053	-5.5945	86.6740
Apr 2020	29.9752	-1.3996	61.3500	-18.0084	77.9588
May 2020	42.1142	9.6295	74.5990	-7.5669	91.7954
Jun 2020	6.8579	-26.2308	39.9467	-43.7470	57.4629
Jul 2020	77.1344	43.3120	110.9567	25.4076	128.8612
Aug 2020	92.2283	57.5087	126.9478	39.1293	145.3273
Sep 2020	-1.5340	-37.0030	33.9351	-55.7792	52.7113
Oct 2020	42.5636	6.4079	78.7193	-12.7317	97.8590
Nov 2020	60.5514	23.6523	97.4506	4.1190	116.9838
Dec 2020	37.7158	0.0849	75.3467	-19.8357	95.2673

```
# 6.4)
predi3<-forecast(fit,h=24)
cbind("predicion" =exp(predi3$mean),"L80" = exp(predi3$lower),"U80" = exp(predi3$uppe
r)) %>%knitr::kable(caption = "Predicciones ")
```

Predicciones

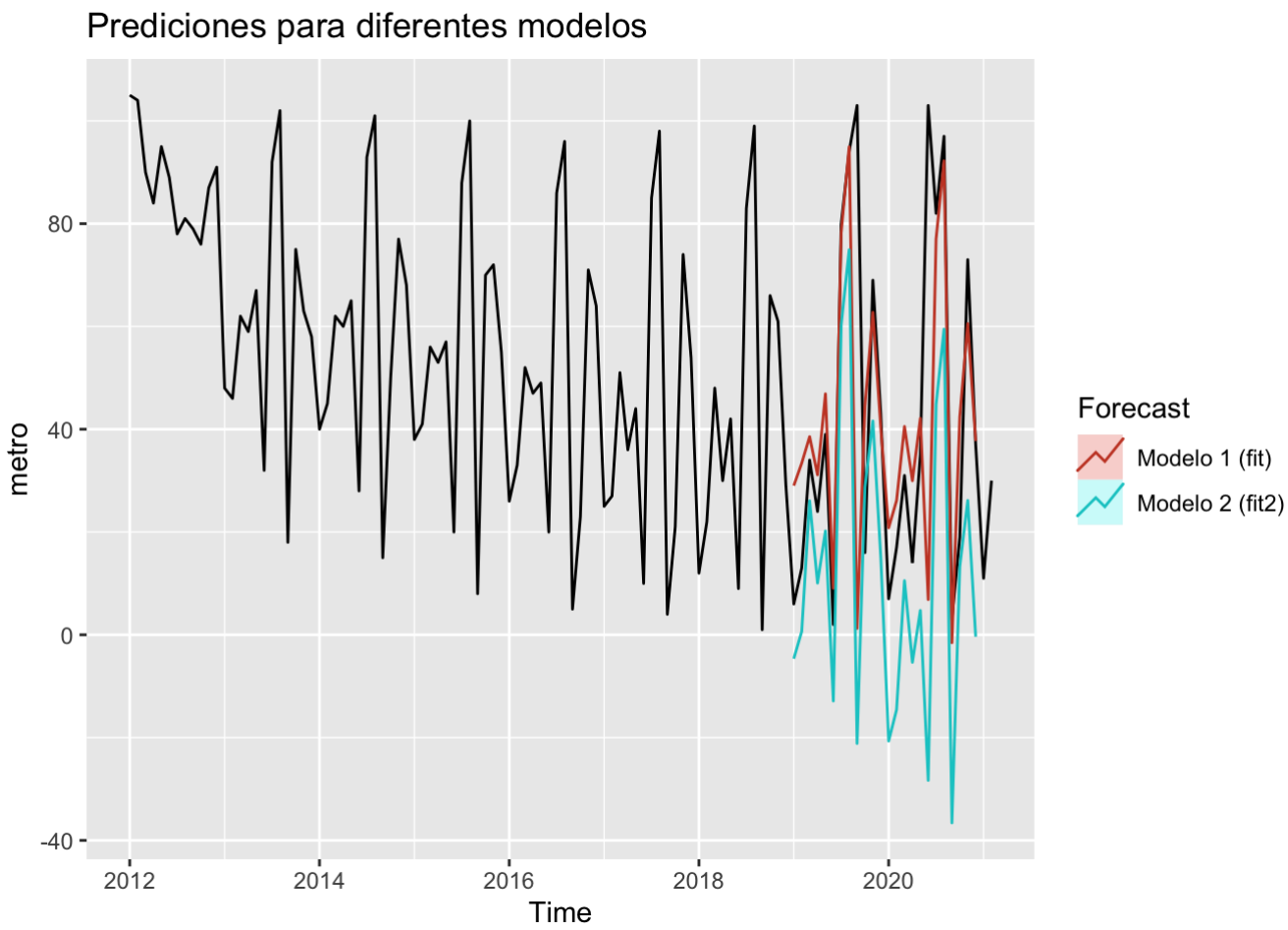
predicion	L80.80%	L80.95%	U80.80%	U80.95%
4.045688e+12	1.775867e+04	6.686117e-01	9.216673e+20	2.447996e+25
3.206321e+14	1.090757e+06	3.588325e+01	9.425105e+22	2.864985e+27
5.556074e+16	1.762834e+08	5.589172e+03	1.751155e+25	5.523171e+29
3.266376e+13	2.484549e+04	3.698593e-01	4.294226e+22	2.884668e+27
2.313390e+20	5.136423e+10	3.984373e+05	1.041926e+30	1.343191e+35
8.591820e+03	1.590164e-06	1.120190e-11	4.642251e+13	6.589899e+18
1.008110e+34	1.243598e+24	7.067457e+18	8.172139e+43	1.437979e+49
1.722712e+41	9.664057e+30	3.618925e+25	3.070903e+51	8.200607e+56
3.483013e+00	1.195891e-10	3.453385e-16	1.014422e+11	3.512896e+16
2.259343e+19	5.280310e+08	1.243876e+03	9.667291e+29	4.103808e+35
1.781298e+27	2.479922e+16	4.440796e+10	1.279484e+38	7.145163e+43
2.986314e+17	2.482768e+06	3.384020e+00	3.591986e+28	2.635347e+34
1.088858e+09	2.286050e-04	4.444456e-11	5.186290e+21	2.667620e+28
2.225656e+11	2.355497e-02	3.186645e-09	2.102972e+24	1.554470e+31
4.038144e+17	3.202127e+04	3.718125e-03	5.092430e+30	4.385708e+37

predicion	L80.80%	L80.95%	U80.80%	U80.95%
1.042462e+13	2.466998e-01	1.510281e-08	4.405059e+26	7.195532e+33
1.949758e+18	1.520634e+04	5.172777e-04	2.499981e+32	7.349156e+39
9.514136e+02	4.056019e-12	1.002164e-19	2.231715e+17	9.032331e+24
3.155259e+33	6.459232e+18	1.082359e+11	1.541307e+48	9.198114e+55
1.133001e+40	9.456210e+24	9.854494e+16	1.357512e+55	1.302646e+63
2.156816e-01	8.507624e-17	5.962395e-25	5.467865e+14	7.801989e+22
3.055903e+18	6.066310e+02	2.955793e-06	1.539411e+34	3.159403e+42
1.982168e+26	1.870865e+10	6.149897e+01	2.100093e+42	6.388707e+50
2.397476e+16	1.088582e+00	2.429160e-09	5.280164e+32	2.366205e+41

Calculando las predicciones y los intervalos de confianza para las unidades de tiempo, podemos concluir que la diferencia entre los valores de pronóstico dados por este modelo y la observación que tuvimos son aceptables.

7. Comparar las predicciones obtenidas con cada uno de los métodos con los valores observados que habíamos reservado antes. Conclusiones.

```
# 7.1)
autoplot(metro) + autolayer(forecast(fit ,h=24), series="Modelo 1 (fit)", PI=FALSE) +
autolayer(forecast(fit2 ,h=24), series="Modelo 2 (fit2)", PI=FALSE) + ggtitle("Predic
iones para diferentes modelos ") + guides(colour=guide_legend(title="Forecast"))
```

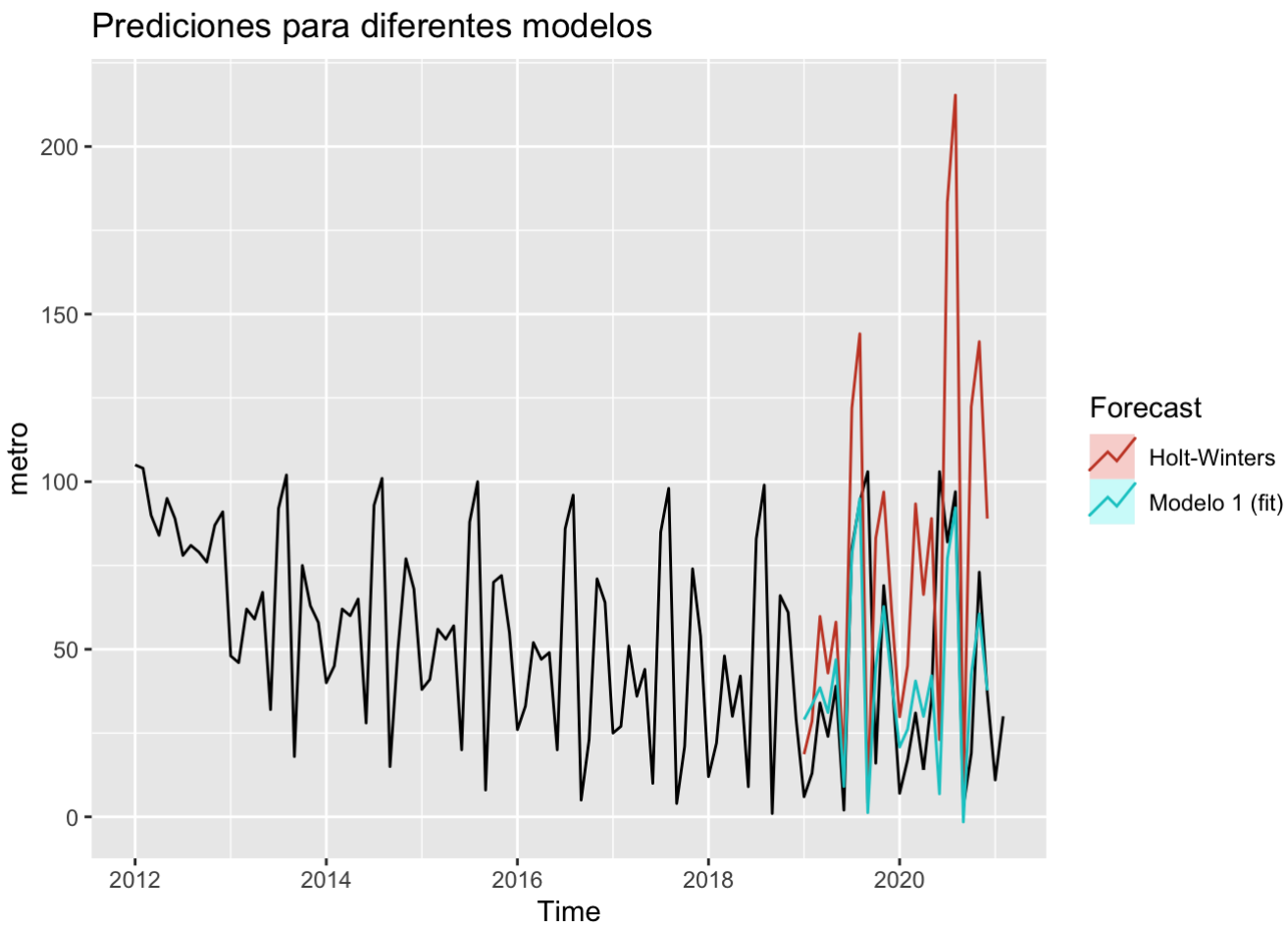


Acabamos de comparar las predicciones obtenidas con cada uno de los métodos, con los valores observados que habíamos reservado antes.

Observamos en esta comparación que el Modelo 1 es el mejor modelo.

A continuación, vamos a comparar el modelo Holt-Winters con el Modelo 1 (fit):

```
# 7.2)
autoplot(metro) + autolayer(forecast(metro_shw ,h=24), series="Holt-Winters", PI=FALS
E) + autolayer(forecast(fit ,h=24), series="Modelo 1 (fit)", PI=FALSE) + ggtitle("Pre
diciones para diferentes modelos ") + guides(colour=guide_legend(title="Forecast"))
```



Según la representación gráfica, observamos de nuevo que el modelo de Arima - Modelo 1 - está más adaptado a los valores de observación de nuestra serie, ya que el Modelo 1 se ajusta mejor que el gráfico de Holt-winters de la serie, sobre todo antes de 2020.

Para terminar, vemos que hay una diferencia importante entre la predicción de ambos modelos en 2020 y los valores observados debido a la pandemia y al número de usuarios de metro en Sevilla que disminuyó por consecuencia.