Series de tiempo univariadas - Presentación 5

Mauricio Alejandro Mazo Lopera

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín

Smoothing Splines:

Originalmente, cuando se desea ajustar una curva a un conjunto de puntos, se busca una función, $g(\cdot)$, tal que la suma cuadrática de los residuales:

$$RSS = \sum_{i=1}^{n} [y_i - g(x_i)]^2$$

sea pequeña.

Smoothing Splines:

Originalmente, cuando se desea ajustar una curva a un conjunto de puntos, se busca una función, $g(\cdot)$, tal que la suma cuadrática de los residuales:

$$RSS = \sum_{i=1}^{n} [y_i - g(x_i)]^2$$

sea pequeña.

En general, se busca que $g(\cdot)$ sea lo más suave posible y pensando en esto se plantea una penalización con respecto a la segunda derivada de $g(\cdot)$ (asumiendo que dicha derivada existe). Este método se conoce como **smoothing spline** y busca minimizar:

$$RSS_\lambda = \sum_{i=1}^{n} [y_i - g(x_i)]^2 + \underbrace{\lambda \int g''(t)^2 dt}_{\textit{penalización}}$$

Smoothing Splines:

En este método λ es un hiperparámetro, que se puede encontrar con validación cruzada, y para el cual se cumple que:

- Si $\lambda = 0$ entonces la penalización no tiene efecto.
- Si $\lambda \longrightarrow \infty$ entonces la penalización es tan grande que $g(\cdot)$ se "suaviza" hasta el punto de convertirse en una línea recta.

Smoothing Splines:

En este método λ es un hiperparámetro, que se puede encontrar con validación cruzada, y para el cual se cumple que:

- Si $\lambda = 0$ entonces la penalización no tiene efecto.
- Si $\lambda \longrightarrow \infty$ entonces la penalización es tan grande que $g(\cdot)$ se "suaviza" hasta el punto de convertirse en una línea recta.

En este método, la función $g(\cdot)$ es spline cúbico natural (antes del primer nodo y después del último se ajustan rectas y en medio un spline cúbico) con nodos en $x_1, x_2, \ldots, x_n$. Esto indica que los grados de libertad de este spline es tan alto como el número de individuos, pero se controlan con el valor de λ, encontrado con validación cruzada Leave-One-Out-Cross-Validation (LOOCV).

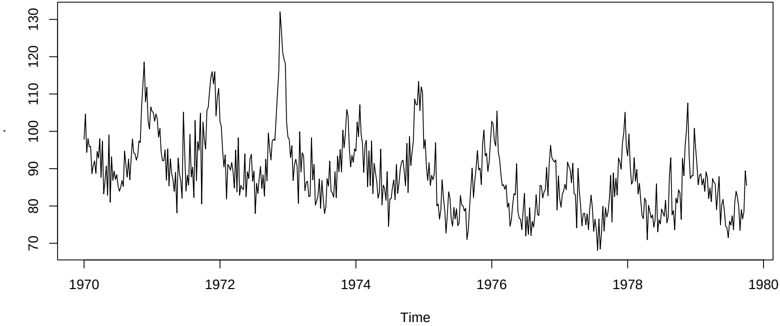
Smoothing Splines:

Este método LOOCV funciona de la siguiente forma:

- Selecciono **distintos valores** de λ por ejemplo valores entre 0 y 10 aumentando de 0.01 en 0.01.
- Para **cada valor fijo** de λ, dejo por fuera cada el primer punto de la serie de tiempo y ajusto el modelo con los demás datos, obteniendo un error cuadrático medio (MSE) de prueba tratando de predecir el dato que dejé por fuera. Repito este procedimiento para el segundo dato, para el tercero, y así sucesivamente hasta el último dato de la serie.
- En el paso anterior obtendo n distintos MSE (uno por cada dato de la serie) y los promedio para encontrar un MSE del λ que fijé anteriormente.
- Repito el proceso para los demás λ y "nos quedamos" con el que arroje el menor MSE obtenido en el paso anterior.

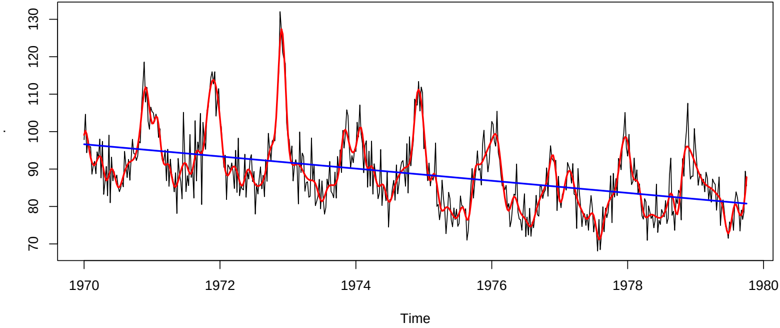
Smoothing Splines:

```
require(astsa)
require(magrittr)
cmort %>% plot()
```



Smoothing Splines:

```r
semana <- cmort %>% time() %>% as.numeric()
modelo4a<-smooth.spline(semana,cmort,lambda=0)
modelo4b<-smooth.spline(semana,cmort,lambda=1000)
cmort %>% plot()
lines(modelo4a,col ="red",lwd =2) # Con lambda=0
lines(modelo4b,col ="blue",lwd =2) # lambda grande
```



Smoothing Splines:

```r
modelo5<-smooth.spline(semana,cmort,cv=TRUE) # Con LOOCV
names(modelo5)
```

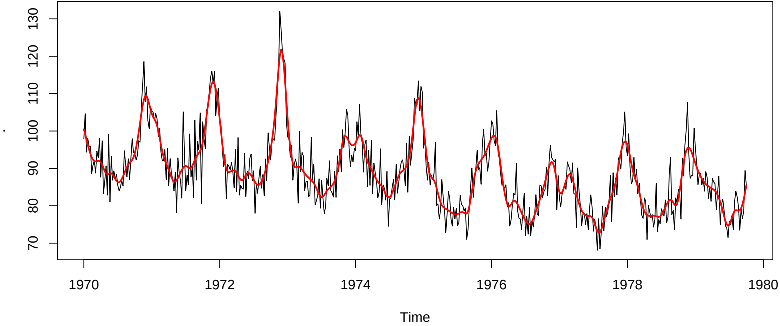
```
##  [1] "x"          "y"          "w"          "yin"        "tol"
##  [6] "data"       "no.weights" "lev"        "cv.crit"    "pen.crit"
## [11] "crit"       "df"         "spar"       "ratio"      "lambda"
## [16] "iparms"     "auxM"       "fit"        "call"
```

```r
modelo5$lambda
```

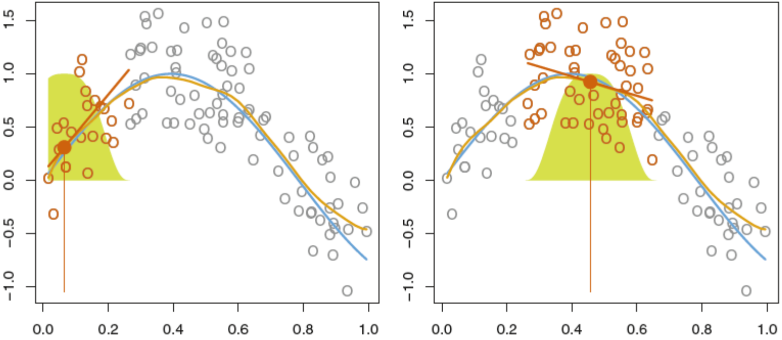
```
## [1] 8.626141e-08
```

Smoothing Splines:

```
cmort %>% plot()
lines(modelo5,col ="red",lwd =2)
```



A pesar de que el smoothing spline describe de manera fiel el comportamiento cíclico de la serie, la mejor opción es modelar la estacionalidad de la misma con técnicas que veremos más adelante.


1.5
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

Se plantea realizar un suavizamiento local alrededor de un punto $t = t_0$. Un algoritmo simple consiste en:

1. Tome una fracción $s = k/n$ de puntos t_i, alrederor de t_0.
2. Dele un peso $K_{i0} = K(t_i, t_0)$ a cada uno de los puntos vecinos de t_0, de tal manera que los vecinos más cercanos tengan mayor peso que los más lejanos. En muchos casos se toma K_{i0} como la función tricubo ($h(t) = (1 - |t|^3)^3$ para $|t| \leq 1$).
3. Estime los parametros β_0 y β_1 minimizando la suma cuadrática ponderada:

$$\sum_{i=1}^{n} K_{i0}(X_i - \beta_0 - \beta_1 Z_i)^2$$

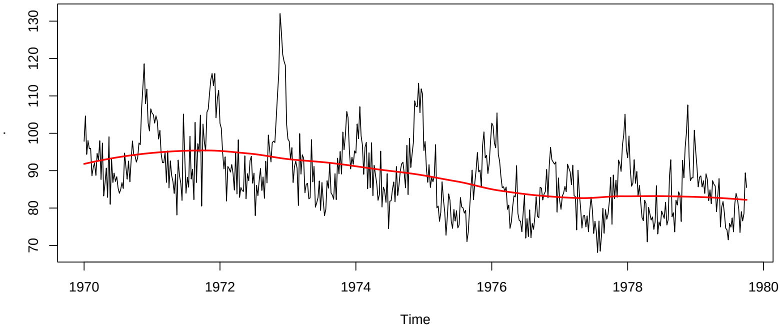
4. El valor ajustado en t_0 y que sirve para elaborar la curva es:

$$\widehat{X}_i = \widehat{\beta_0} + \widehat{\beta}_1 t_0$$

Regresión local:

```
modelo6<-loess(cmort~semana, span=0.5)
```

```
cmort %>% plot()
lines(semana, modelo6$fitted,col ="red",lwd =2)
```



¿Qué pasa si aumentamos o disminuimos el hiperparámetro **span**?

¿Para qué ajustamos la tendencia T_t?

Los métodos que vimos anteriormente se usan para:

- Observar una curva suave que elimine el "ruido" de la serie y así identificar tendencias o comportamientos estacionales o cíclicos.

¿Para qué ajustamos la tendencia T_t?

Los métodos que vimos anteriormente se usan para:

- Observar una curva suave que elimine el "ruido" de la serie y así identificar tendencias o comportamientos estacionales o cíclicos.
- Modelar o ajustar la tendencia, $\widehat{T}_t$, para eliminarla y dejar una nueva serie, $Y_t = X_t - \widehat{T}_t$, a la cual se le pueden seguir aplicando métodos para explicar estacionalidad, su dependencia con valores pasados, transformaciones para estabilizar la varianza, etc.

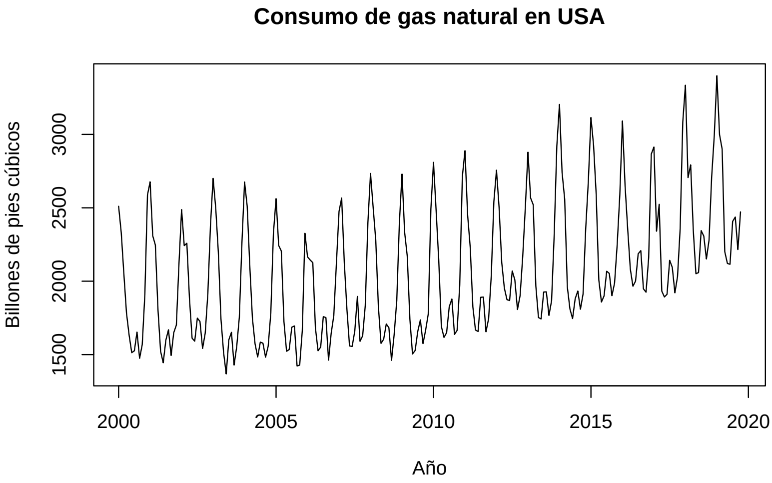
NOTA: Más adelante veremos otra estrategia para eliminar la tendencia de una serie mediante diferenciación discreta.

¿Qué alternativas existen para la parte estacional S_t?

Una alternativa consiste en ajustar un modelo con una variable categórica que represente los periodos del ciclo estacional. Por ejemplo, si existe un ciclo estacional año tras año, entonces creamos una variable categórica que sea mes y con esta ajustamos un modelo.

Ejemplo: Modelando la parte estacional S_t:

```r
require(TSstudio)
plot(USgas, main = "Consumo de gas natural en USA",
     ylab= "Billones de pies cúbicos",xlab= "Año")
```



Ejemplo: Modelando la parte estacional S_t:

Apliquemos la función **decompose** del paquete **stats**:

```
USgas_decompose <- decompose(USgas, type="additive")
str(USgas_decompose)
```

```
## List of 6
##  $ x       : Time-Series [1:238] from 2000 to 2020: 2510
##  $ seasonal: Time-Series [1:238] from 2000 to 2020: 766
##  $ trend   : Time-Series [1:238] from 2000 to 2020: NA N
##  $ random  : Time-Series [1:238] from 2000 to 2020: NA N
##  $ figure  : num [1:12] 766 453 278 -174 -352 ...
##  $ type    : chr "additive"
##  - attr(*, "class")= chr "decomposed.ts"
```

```
class(USgas_decompose)
```

```
## [1] "decomposed.ts"
```

Ejemplo: Modelando la parte estacional S_t:

La anterior descomposición se aplica solo a objetos **ts** y los resultados obtenidos se pueden analizar según lo siguiente:

La anterior descomposición se aplica solo a objetos **ts** y los resultados obtenidos se pueden analizar según lo siguiente:

- **x**: es la serie de tiempo original (objeto **ts**).

Ejemplo: Modelando la parte estacional S_t:

La anterior descomposición se aplica solo a objetos **ts** y los resultados obtenidos se pueden analizar según lo siguiente:

- **x**: es la serie de tiempo original (objeto **ts**).
- **trend** ($\widehat{T}_t$): es el estimado de la tendencia obtenida con medias móviles bilaterales descrito en la diapositiva 13 de la Presentación 2 del curso. Por este motivo aparecen un NA al principio y al final.

Ejemplo: Modelando la parte estacional S_t:

La anterior descomposición se aplica solo a objetos **ts** y los resultados obtenidos se pueden analizar según lo siguiente:

- **x**: es la serie de tiempo original (objeto **ts**).
- **trend** ($\widehat{T}_t$): es el estimado de la tendencia obtenida con medias móviles bilaterales descrito en la diapositiva 13 de la Presentación 2 del curso. Por este motivo aparecen un NA al principio y al final.
- **seasonal** ($\widehat{S}_t$): Es el componente estacional obtenido luego de eliminar la tendencia. Tenga en cuenta que depende de la frecuencia que esté utilizando el objeto **ts** (use **ts_info(USgas)** para obtener la información en este ejemplo).

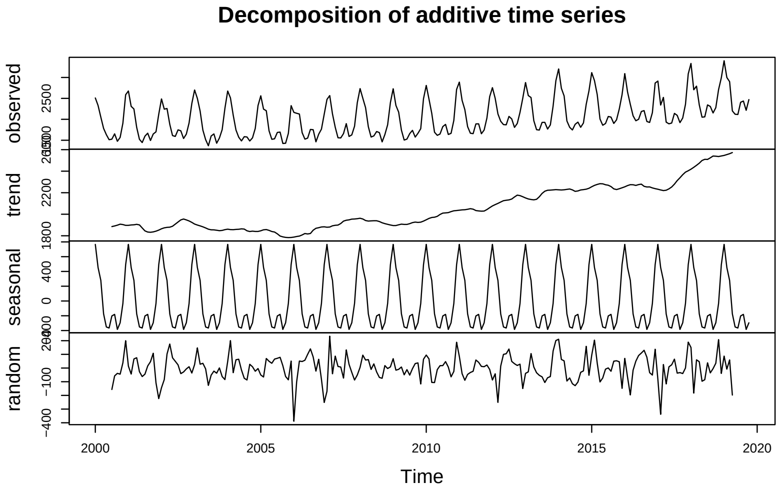
Ejemplo: Modelando la parte estacional S_t:

La anterior descomposición se aplica solo a objetos **ts** y los resultados obtenidos se pueden analizar según lo siguiente:

- **x**: es la serie de tiempo original (objeto **ts**).
- **trend** ($\widehat{T}_t$): es el estimado de la tendencia obtenida con medias móviles bilaterales descrito en la diapositiva 13 de la Presentación 2 del curso. Por este motivo aparecen un NA al principio y al final.
- **seasonal** ($\widehat{S}_t$): Es el componente estacional obtenido luego de eliminar la tendencia. Tenga en cuenta que depende de la frecuencia que esté utilizando el objeto **ts** (use **ts_info(USgas)** para obtener la información en este ejemplo).
- **random** ($\widehat{I}_t$): Corresponde al componente irregular que aparece luego de extraer las componentes $\widehat{T}_t$ y $\widehat{S}_t$ de la serie.
- **type**: Las opciones son **additive** o **multiplicative**.

Ejemplo: Modelando la parte estacional S_t:

```
plot(USgas_decompose)
```



Ejemplo: Modelando la parte estacional S_t:

Para modelar la parte estacional, debemos crear un data frame:

```
USgas_df <- data.frame(anio_mes=as.numeric(time(USgas)),
                       mes=cycle(USgas),
                       valores = as.numeric(USgas),
                       t = 1:length(USgas))
```

Ejemplo: Modelando la parte estacional S_t:

Para modelar la parte estacional, debemos crear un data frame:

```r
USgas_df <- data.frame(anio_mes=as.numeric(time(USgas)),
                       mes=cycle(USgas),
                       valores = as.numeric(USgas),
                       t = 1:length(USgas))
```

Transformamos en factor el mes:

```r
USgas_df$mes <- factor(month.abb[USgas_df$mes],
                       levels=month.abb)
```

Ejemplo: Modelando la parte estacional S_t:

Para modelar la parte estacional, debemos crear un data frame:

```
USgas_df <- data.frame(anio_mes=as.numeric(time(USgas)),
                       mes=cycle(USgas),
                       valores = as.numeric(USgas),
                       t = 1:length(USgas))
```

Transformamos en factor el mes:

```
USgas_df$mes <- factor(month.abb[USgas_df$mes],
                       levels=month.abb)
```

```
head(USgas_df, n=3)
```

```
##   anio_mes mes valores t
## 1 2000.000 Jan  2510.5 1
## 2 2000.083 Feb  2330.7 2
## 3 2000.167 Mar  2050.6 3
```

Ejemplo: Modelando la parte estacional S_t:

El modelo aditivo que considera modelar la tendencia y la parte estacional es:

```
mod_tend_est1 <- lm(valores ~ t + mes, data=USgas_df)
```

En este modelo estamos ajustando los **valores** de la serie a través de una regresión múltiple donde:

- La tendencia (T_t) está representada por la variable **t**.
- La estacionalidad o ciclos estacionales (S_t) está representada por la variables **mes**).

Ejemplo: Modelando la parte estacional S_t:

```
summary(mod_tend_est1)
```

```
##
## Call:
## lm(formula = valores ~ t + mes, data = USgas_df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -518.22  -83.95  -20.94   88.73  380.74
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2471.3920    33.8984  72.906  < 2e-16 ***
## t               2.9058     0.1288  22.568  < 2e-16 ***
## mesFeb       -305.9658    43.1244  -7.095 1.67e-11 ***
## mesMar       -485.9566    43.1250 -11.269  < 2e-16 ***
## mesApr       -928.6724    43.1260 -21.534  < 2e-16 ***
## mesMay      -1109.5632    43.1273 -25.728  < 2e-16 ***
## mesJun      -1128.9840    43.1290 -26.177  < 2e-16 ***
## mesJul       -959.4349    43.1312 -22.245  < 2e-16 ***
## mesAug       -936.8907    43.1337 -21.721  < 2e-16 ***
## mesSep      -1141.4665    43.1365 -26.462  < 2e-16 ***
## mesOct      -1043.6273    43.1398 -24.192  < 2e-16 ***
## mesNov       -801.8464    43.6910 -18.353  < 2e-16 ***
## mesDec       -277.4575    43.6927  -6.350 1.17e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 136.4 on 225 degrees of freedom
## Multiple R-squared:  0.909,  Adjusted R-squared:  0.9042
## F-statistic: 187.3 on 12 and 225 DF,  p-value: < 2.2e-16
```

Ejemplo: Modelando la parte estacional S_t:

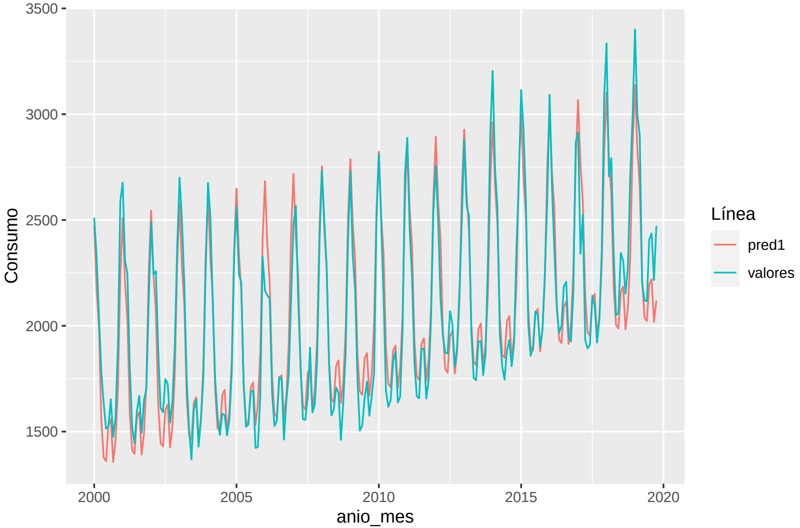
Preparamos un nuevo data frame para poder comparar los valores estimados con los valores reales:

```r
require(tidyverse)
require(magrittr)
USgas_df$pred1 <- predict(mod_tend_est1, newdata=USgas_df)
USgas_ajus1 <- USgas_df %>%
  gather(key="Línea", value="Consumo", valores, pred1)
```

Ahora realizamos el gráfico con los códigos:

```r
USgas_ajus1 %>% ggplot(aes(x=anio_mes, y=Consumo,
                           col=Línea))+
  geom_line()
```

Ejemplo: Modelando la parte estacional S_t:



Ejemplo: Modelando la parte estacional S_t:

También es posible ajustar un modelo polinomial cuadrático para ajustar la tendencia y la estacionalidad:

```r
mod_tend_est2 <- lm(valores ~ poly(t, degree=2) + mes,
                    data=USgas_df)
```

Ejemplo: Modelando la parte estacional S_t:

También es posible ajustar un modelo polinomial cuadrático para ajustar la tendencia y la estacionalidad:

```
mod_tend_est2 <- lm(valores ~ poly(t, degree=2) + mes,
                    data=USgas_df)
```

Creamos nuevamente un data frame para poder contrastar gráficamente el modelo cuadrático más la estacionalidad con los valores reales:

```
USgas_df$pred2 <- predict(mod_tend_est2, newdata=USgas_df)
USgas_ajus2 <- USgas_df %>%
  gather(key="Línea", value="Consumo", valores, pred2)
```

Ejemplo: Modelando la parte estacional S_t:

También es posible ajustar un modelo polinomial cuadrático para ajustar la tendencia y la estacionalidad:

```
mod_tend_est2 <- lm(valores ~ poly(t, degree=2) + mes,
                    data=USgas_df)
```

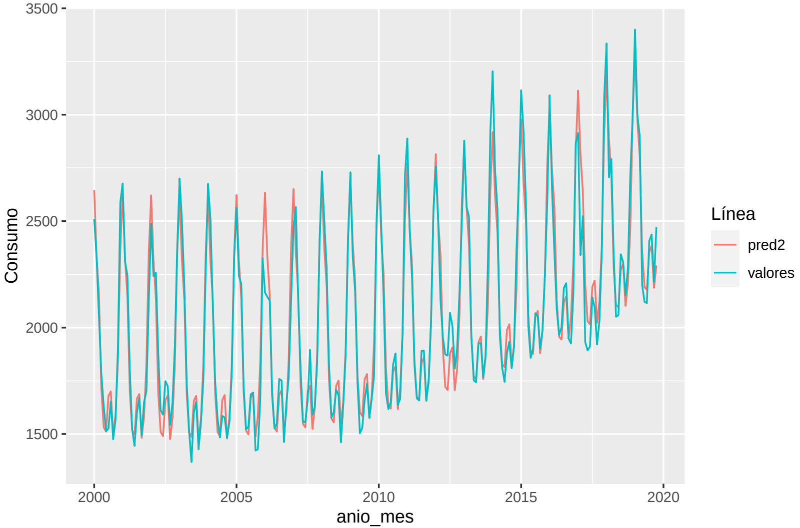
Creamos nuevamente un data frame para poder contrastar gráficamente el modelo cuadrático más la estacionalidad con los valores reales:

```
USgas_df$pred2 <- predict(mod_tend_est2, newdata=USgas_df)
USgas_ajus2 <- USgas_df %>%
  gather(key="Línea", value="Consumo", valores, pred2)
```

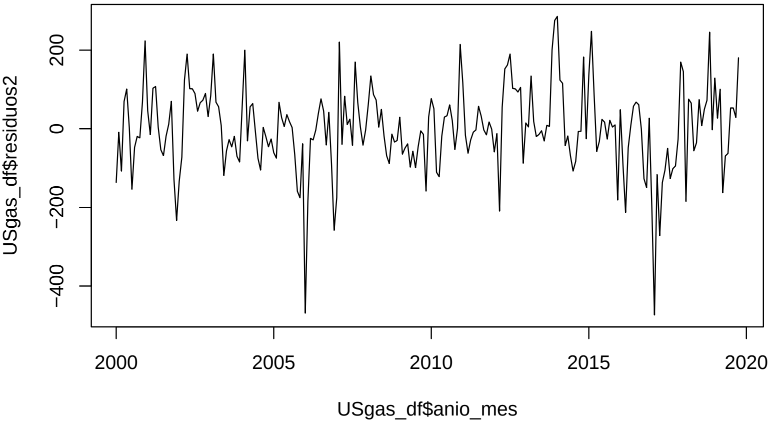
Ahora realizamos el gráfico con los códigos:

```
USgas_ajus2 %>% ggplot(aes(x=anio_mes, y=Consumo,
                           col=Línea))+
  geom_line()
```

Ejemplo: Modelando la parte estacional S_t:

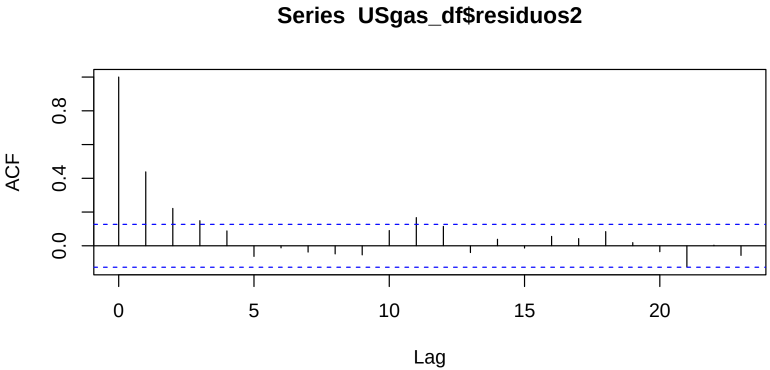


Los residuales de este último modelo se obtienen como:



Ejemplo: Modelando la parte estacional S_t:

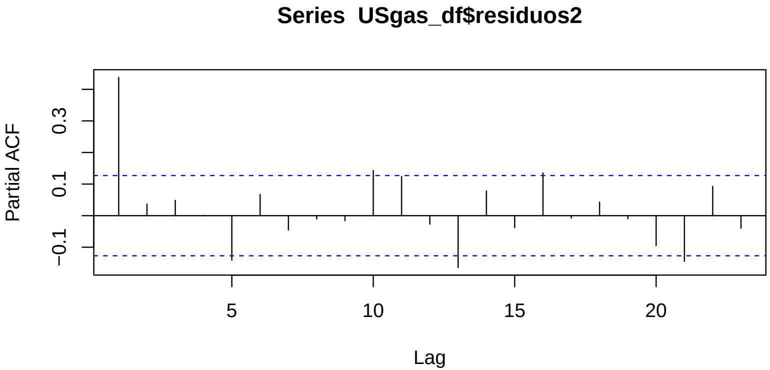
Las ACF muestral de los residuales anteriores está dada por:



Como algunas de las correlaciones muestrales se salen de la banda horizontal, entonces aún no se pueden considerar como ruido blanco y esto hace necesario explorar otros modelos de series de tiempo para estos residuales.

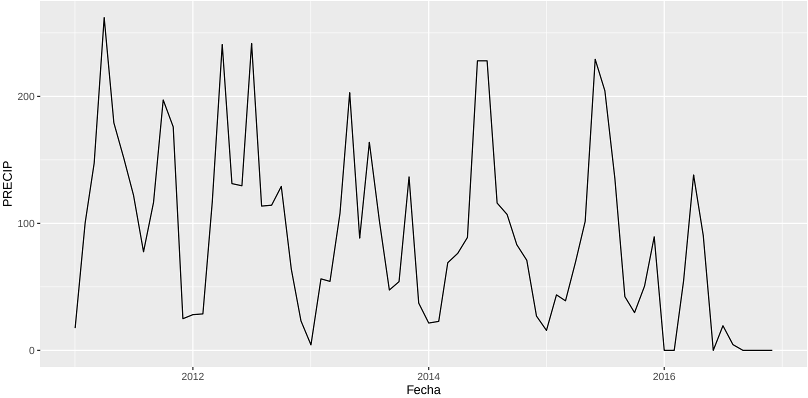
Ejemplo: Modelando la parte estacional S_t:

Las PACF muestral de los residuales anteriores está dada por:



Como en el comentario de la diapositiva anterios, aparentemente es necesario explorar otros modelos de series de tiempo para estos residuales.

Ejemplo: Precipitaciones - El Local

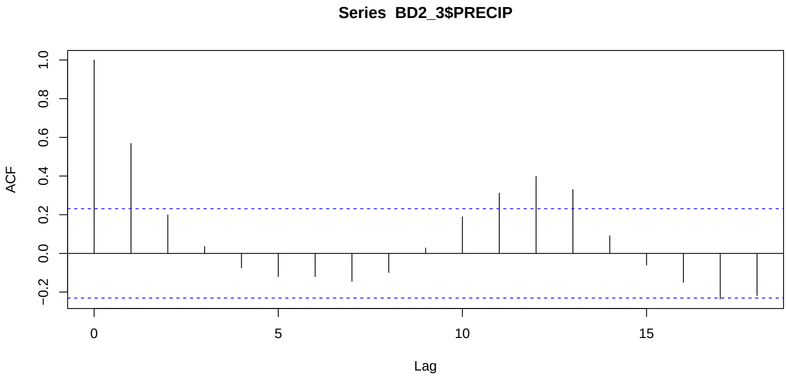


CÓDIGOS

Esta serie tiene las siguientes gráficas de autocorrelación muestral:

Ejemplo: Precipitaciones - El Local

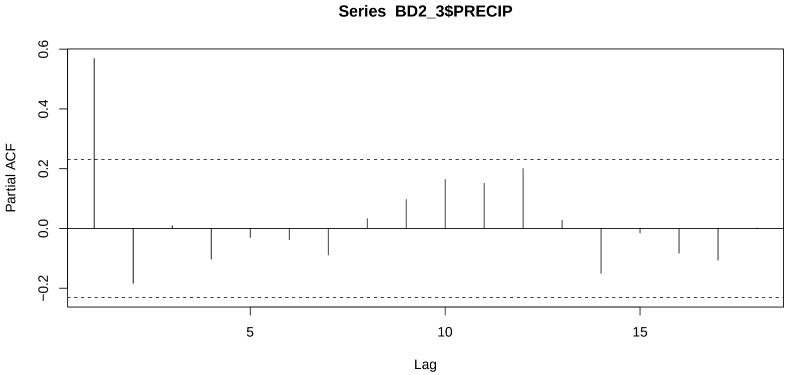
```
acf(BD2_3$PRECIP)
```



Se observa presencia de ciclos estacionales dado por los meses.

Ejemplo: Precipitaciones - El Local

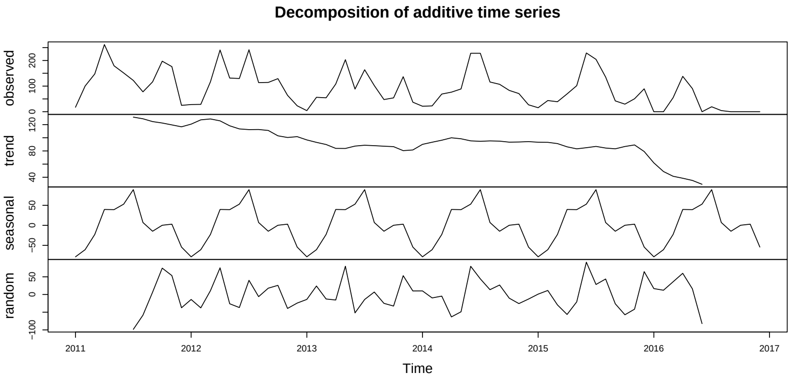
```
pacf(BD2_3$PRECIP)
```



Se observa algo parecido al gráfico anterior pero con correlaciones parciales mayormente no significativas.

Ejemplo: Precipitaciones - El Local

```
local_ts <- ts(BD2_3$PRECIP,start =2011, frequency = 12 )
local_decompose <- decompose(local_ts)
plot(local_decompose)
```



Ejemplo: Precipitaciones - El Local

Como en la serie original no se evidencia una tendencia, entonces ajustamos un modelo solo para la estacionalidad (S_t):

```r
mod_local <- lm(PRECIP~t+MES, data=BD2_3)
summary(mod_local)
```

```
##
## Call:
## lm(formula = PRECIP ~ t + MES, data = BD2_3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -101.139  -34.540   -2.989   26.937  117.083
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    197.5199    22.5079   8.776 2.71e-12 ***
## t               -1.4231     0.2859  -4.977 5.92e-06 ***
## MESAGOSTO      -51.9241    28.7312  -1.807 0.075824 .
## MESDICIEMBRE  -104.1316    28.7994  -3.616 0.000621 ***
## MESENERO      -138.8861    28.7212  -4.836 9.87e-06 ***
## MESFEBRERO    -110.0129    28.7141  -3.831 0.000311 ***
## MESJULIO        18.3694    28.7212   0.640 0.524924
## MESJUNIO        -8.5537    28.7141  -0.298 0.766831
## MESMARZO       -70.3898    28.7098  -2.452 0.017195 *
## MESMAYO        -15.3269    28.7098  -0.534 0.595448
## MESNOVIEMBRE   -56.1547    28.7781  -1.951 0.055775 .
## MESOCTUBRE     -58.3779    28.7596  -2.030 0.046889 *
## MESSEPTIEMBRE  -70.7510    28.7440  -2.461 0.016782 *
## ---
```

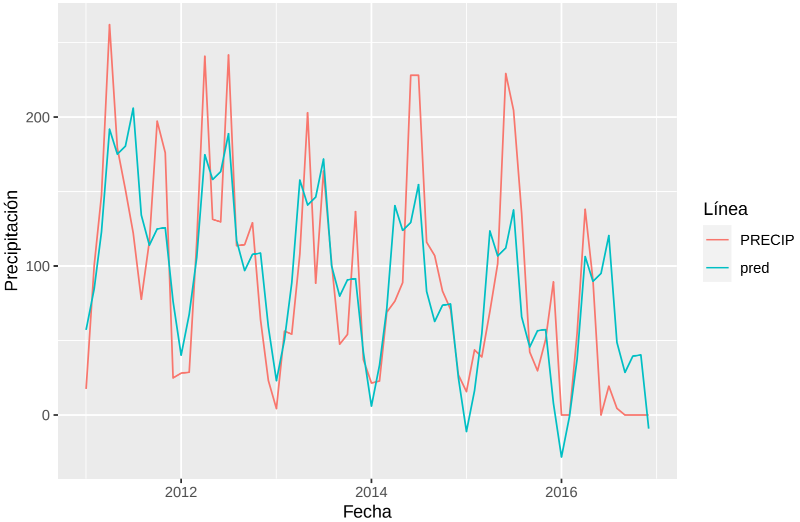
Obtenemos los valores ajustados del modelo:

```
BD2_3$pred <- predict(mod_local, newdata=BD2_3)
```

Obtenemos un nuevo data frame para visualizar las predicciones del modelo con los valores reales:

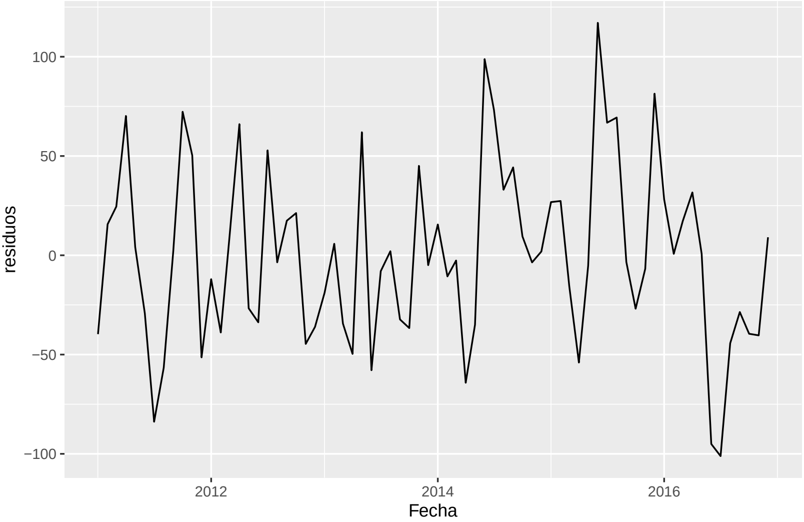
```
BD2_3_ajus <- BD2_3 %>%
  gather(key="Línea", value="Precipitación",
         PRECIP, pred)
```

Ejemplo: Precipitaciones - El Local



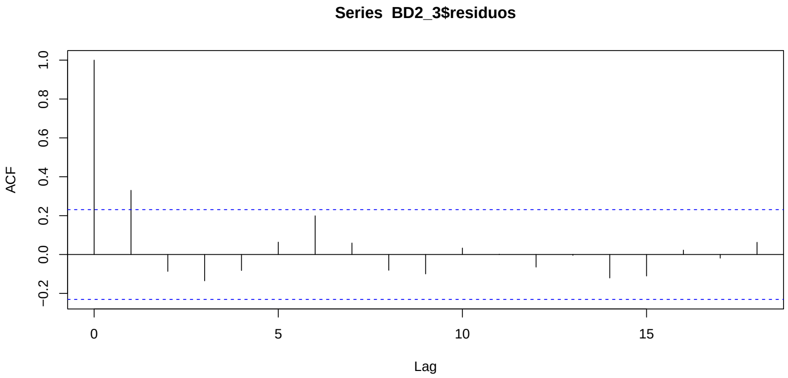
Ejemplo: Precipitaciones - El Local

Los residuales de este último modelo se obtienen como:



Ejemplo: Precipitaciones - El Local

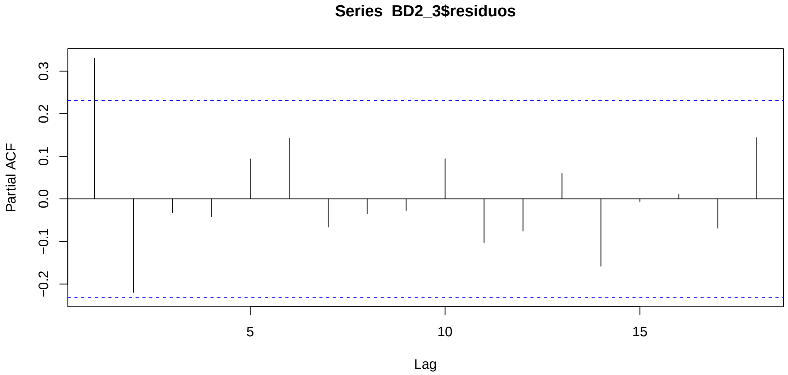
```
acf(BD2_3$residuos)
```



Solo se observa la primera correlación por fuera de la banda de confianza y no los ciclos estacionales.

Ejemplo: Precipitaciones - El Local

```
pacf(BD2_3$residuos)
```



Solo se observa la primera correlación por fuera de la banda de confianza y los demás dentro.

Conclusiones:

- La componente de tendencia, T_t, se puede modelar de forma clásica utilizando algún método de ajuste clásico de regresión lineal dando como covariable una variable t que va desde el 1 hasta la longitud de la serie.

Conclusiones:

- La componente de tendencia, T_t, se puede modelar de forma clásica utilizando algún método de ajuste clásico de regresión lineal dando como covariable una variable t que va desde el 1 hasta la longitud de la serie.
- La componente de ciclos estacionales o simplemente estacional, S_t, se puede modelar creando un factor que se utilice para modelas los valores de la serie.

Conclusiones:

- La componente de tendencia, T_t, se puede modelar de forma clásica utilizando algún método de ajuste clásico de regresión lineal dando como covariable una variable t que va desde el 1 hasta la longitud de la serie.
- La componente de ciclos estacionales o simplemente estacional, S_t, se puede modelar creando un factor que se utilice para modelas los valores de la serie.
- Una vez ajustadas la tendencia y la estacionalidad, se deben analizar los residuos del modelo en el orden cronológico de la serie de tiempo y ver si son ruido blanco o no.

Conclusiones:

- La componente de tendencia, T_t, se puede modelar de forma clásica utilizando algún método de ajuste clásico de regresión lineal dando como covariable una variable t que va desde el 1 hasta la longitud de la serie.
- La componente de ciclos estacionales o simplemente estacional, S_t, se puede modelar creando un factor que se utilice para modelas los valores de la serie.
- Una vez ajustadas la tendencia y la estacionalidad, se deben analizar los residuos del modelo en el orden cronológico de la serie de tiempo y ver si son ruido blanco o no.
- En lo que viene del curso veremos otros modelos para ajustar tendencia, estacionalidad y autocorrelación de una serie de tiempo, distintos a los descritos en esta presentación.


MUCHAS
Gracias!

Anexos - El Local

```r
BD2<-read.csv("../../DATOS/Precipitaciones_Totales_Mensuales.csv",
                header=TRUE,fileEncoding = "utf8")
BD2_1 <- filter(BD2, ESTACION=="El Local", ANIO>2010)
BD2_2 <- gather(BD2_1, "ENERO", "FEBRERO", "MARZO", "ABRIL",
                "MAYO", "JUNIO", "JULIO","AGOSTO",
                "SEPTIEMBRE","OCTUBRE", "NOVIEMBRE",
                "DICIEMBRE", key="MES",value="PRECIP")

BD2_3 <- BD2_2[order(BD2_2$ANIO),]
BD2_3$t <- 1:nrow(BD2_2)
BD2_3$Fecha <- paste(rep(1,nrow(BD2_3)),
                     BD2_3$MES,BD2_3$ANIO)

BD2_3$Fecha %<>% as.Date(format="%d %B %Y")
ggplot(BD2_3, aes(x=Fecha, y=PRECIP))+
  geom_line(col="black")
```

VOLVER