Modelización de series temporales: aplicaciones en R

Autor: Jose González Abad
Tutora: Rosa Sepúlveda Correa

Universidad de Salamanca

19 de Julio, 2017

Índice

Objetivos

- Introducción teórica el análisis de series temporales.
- Recorrido por librerías de R orientadas a:
  - Estructurar los datos.
  - Modelizar.
- Conclusiones.
  - Metodología para el análisis de series temporales en R.

Enfoque determinista

Definición

Supone que la serie temporal se puede expresar como la combinación de distintas componentes.

$$Y_t = f(t) + a_t \quad \text{con} \quad a_t \approx RB(0, \sigma^2)$$

Componentes

- **Tendencia** (T_t): Evolución de la serie a largo plazo.
- **Estacionalidad** (S_t): Variaciones recurrentes en periodos de tiempo cortos.
- **Ciclo** (C_t): Variaciones recurrentes en periodos de tiempo largos.
- **Fluctuaciones irregulares** (a_t): Fluctuaciones con apariencia no determinística.

Estructura

- **Aditiva**:

$$Y_t = T_t + S_t + C_t + a_t$$

- **Multiplicativa**:

$$Y_t = T_t \times S_t \times C_t \times a_t$$

Estimación de las componentes

- **Estimación de la Tendencia**
  - Ajuste de funciones a la tendencia (lineal, exponencial...).
  - Método de la media móvil:

$$Y_t' = \frac{1}{p}\left(Y_{t-\left(\frac{p-1}{2}\right)} + ... + Y_{t-1} + Y_t + Y_{t+1} + ... + Y_{t+\left(\frac{p-1}{2}\right)}\right)$$

- **Estimación del ciclo**: Esta componente es difícil de estimar. Se suele incluir junto a la tendencia.
- **Estimación de la estacionalidad**:
  - Ajuste de funciones periódicas a la estacionalidad.
  - Variables *dummy*:

$$g(t) = \sum_{i=1}^{S}(\lambda_i d_{it})$$

- **Estimación de la componente irregular**:

$$Y_t - T_t - S_t - C_t = a_t$$

$$\frac{Y_t}{T_t \times S_t \times C_t} = a_t$$

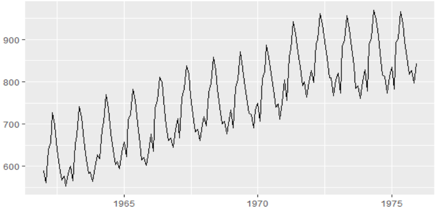


Figura: Serie original

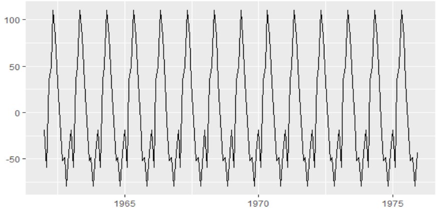


Figura: Estacionalidad

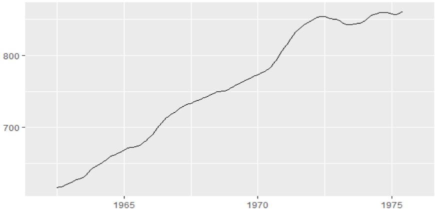


Figura: Tendencia

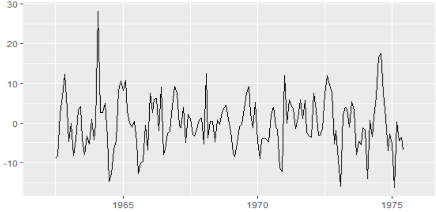


Figura: Componente irregular

Metodología Box-Jenkins

Definición

Supone que la serie temporal es la realización de un proceso estocástico determinado y modelizable. Se sustenta en los modelos ARIMA.

Estacionariedad débil

Para trabajar con esta familia de modelos es necesario trabajar bajo los siguientes supuestos:

$$E[Y_t] = \mu < \infty \qquad \forall t$$

$$Var(Y_t) = \sigma^2 < \infty \qquad \forall t$$

$$Cov(Y_t, Y_{t+k}) = E[(Y_t - \mu)(Y_{t+k} - \mu)] < \infty \qquad \forall t$$

Modelos ARIMA

Suponen que la serie en un momento t depende de sus valores pasados hasta el rezago $t-p$, de su innovación contemporánea y de las pasadas hasta el rezago $t-q$:

$$Y_t = a_t + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} (-\theta_i) a_{t-i}$$

En términos del operador de retardos se puede expresar como:

$$\phi_p(L) Y_t = \theta_q(L) a_t$$

La serie no es estacionaria si la parte autorregresiva posee alguna raíz unitaria (d):

$$\phi_p(L) Y_t = \gamma_{p-d}(L) \cdot (1-L)^d$$

Para solucionar esto se diferencia la serie d veces:

$$\gamma_{p-d}(L) \cdot \Delta^d Y_t = \theta_q(L) a_t$$

- **Preparación de los datos**: Buscamos que nuestra serie sea estacionaria tanto en tendecia como en varianza.
  - Estacionariedad en tendencia: Diferenciamos la serie d veces, hasta conseguir estacionariedad.
  - Estacionariedad en varianza: Estabilizamos la serie con logaritmos o transformación Box-Cox
- **Identificación y selección del modelo**: Nuestro objetivo es encontrar los valores óptimos de p y q. Para ello debemos estudiar la evolución de la función de autocorrelación y compararla con la teórica de los distintos modelos.
- **Estimación de los parámetros**: Método de máxima verosimilitud, mínimos cuadrados no lineales...
- **Validación del modelo**: Comprobamos si el ajuste de nuestro modelo es efectivamente el correcto. Para ello podemos estudiar:
  - Distribución y correlaciones de los residuos
  - Significativadad de los coeficientes
  - Poder predictivo
- **Predicción**: Realizamos las predicciones para Y_{t+h} con el modelo ajustado y validado.

Estructura de los datos temporales en R

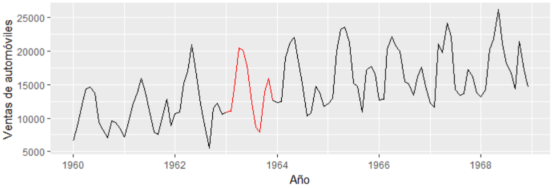
Este tipo de datos necesitan clases especiales para el correcto tratamiento, no podemos recurrir al `data.frame` clásico.

Librería `stats`

- Con la librería `stats` es posible definir objetos de clase `ts`.

```
ts(data = NA, start = 1, end = numeric(), frequency = 1, ...)
```

- Además de otros métodos enfocados a otras tareas:
  - `plot.ts()` para representar gráficamente estos objetos.
  - `cbind()` para agrupar varias series en un mismo objeto.
  - `window()` para seleccionar fácilmente un subconjunto de la serie.



Librería zoo

- Esta clase de objetos nos permite una estructuración más avanzada de series temporales. Se adaptan a periodos temporales irregulares.

```
zoo(x = NULL, order.by = index(), frequency = NULL)
```

- Métodos como `index()`, `coredata()` y `merge.zoo()` nos facilitan la tarea de trabajar con la clase `zoo`.
- Esta librería incluye métodos muy potentes para el tratamiento de `NAs`:
  - `na.approx()`: recurre a interpolaciones lineales.
  - `na.spline()`: recurre a interpolaciones cúbicas de splines.
  - `na.locf()`: utiliza valores cercanos al valor faltante.
- El método `rollapply()` nos permite aplicar funciones a subconjuntos sucesivos de la serie.

```
rollapply(data, width, FUN, align)
```

- El método `rollmean()` aplica un suavizado de medias móviles a nuestra serie. Es un implementación más directa del anterior.

Librería `xts`

- Define la clase `xts`, la cual se deriva de `zoo`.

```
xts(x = NULL, order.by = index(x), frequency = NULL, unique = TRUE,...)
```

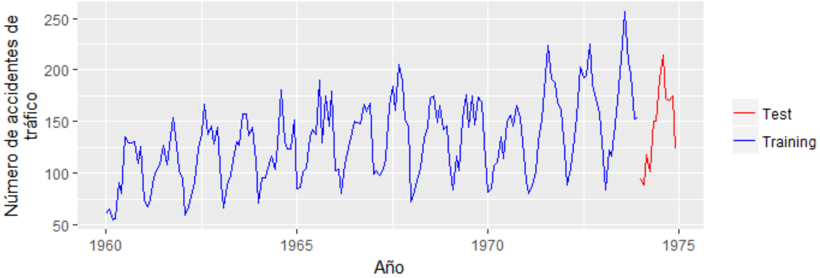
- Esta clase implementa una conversión más potente entre clases a través de `as.xts()`.
- Implementa una selección muy intuitiva de observaciones a través de fechas.

```
# diciembre de 1963
xts.sales["1963-12"]
# el año completo de 1963
xts.sales["1963"]
# todas las observaciones hasta julio de 1963
xts.sales["/1963-7"]
# todas las observaciones a partir de julio de 1963
xts.sales["1963-7/"]
# observaciones comprendidas entre julio de 1962 y de 1963
xts.sales["1962-7/1963-7"]
```

- El método `period.apply()` realiza una función similar al `rollapply()` de `zoo`.

Modelización de series temporales con R

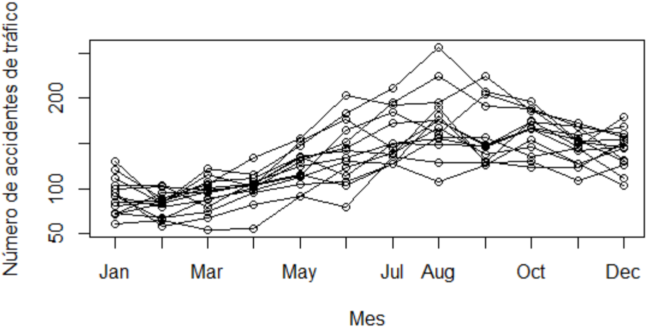
- Trabajaremos con la serie correspondiente al número mensual de accidentes de tráfico ocurridos en Ontario en el periodo 1969-1974
- Dividiremos la serie en un conjunto de *training* (1969-1973) y en uno de *test* (1974).



Forecast package

Esta librería está enfocada tanto al preprocesado como al análisis de series temporales univariantes. Trabajaremos con la clase `ts`.

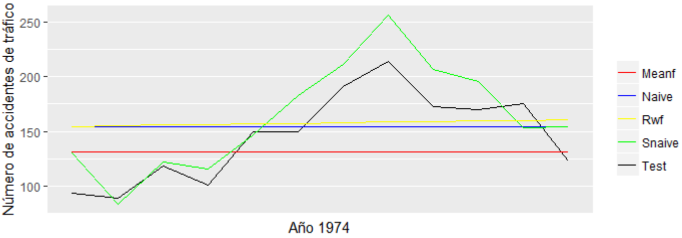
- Para estudiar la estacionalidad más en detalle podemos recurrir al método `seasonplot()`.



Modelos ingenuos

Con esta librería es posible ajustar varios modelos ingenuos a nuestros datos. Esto modelos nos pueden servir para trabajar bajo el principio de parsimonia.

- `meanf()`: Realiza predicciones con la media de la serie
- `naive()`: Predice con el valor anterior observado.
- `rwf()`: Variación de `naive()` con pendiente.
- `snaives()`: Predice con el correspondiente valor estacional anterior.



El método `accuracy()` nos ofrece medidas de precisión del modelo en el conjunto de *training* y *test*. Si lo aplicamos sobre el modelo de `meanf()` y el conjunto de *test* `acc.test` obtenemos lo siguiente:

```
                        ME     RMSE      MAE        MPE     MAPE     MASE
Training set 5.392491e-15 38.73302 31.48058 -10.085477 27.71539 1.774836
Test set     1.439286e+01 41.43521 36.28770   2.639993 25.15689 2.045855
                  ACF1 Theil's U
             0.7359523        NA
             0.6310518  1.205019
```

Para evaluar los modelos usaremos principalmente el MAE:

$$MAE = \frac{1}{T}\sum_{i=1}^{n}|Y_i - \widehat{Y}_i|$$

El mejor MAE en el *test* le obtenemos con `snaive()` (22.58333).
Necesitamos ajustar modelos más complejos.

Modelo lineal

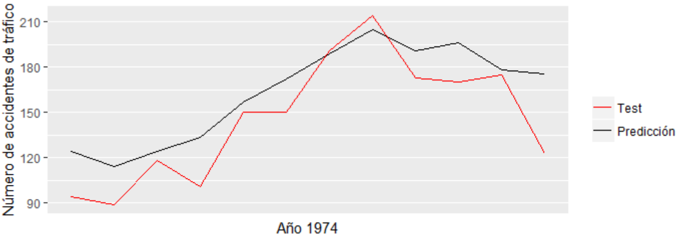
El método `tslm()` ajusta el siguiente modelo lineal:

$$Y_t = \beta_0 + \beta_1 t + \sum_{i=2}^{M} \lambda_i d_i$$

Si lo aplicamos a nuestros obtenemos el siguiente modelo:

$$\widehat{Y}_t = 63{,}75 + 0{,}35t - 9{,}99d_2 - 0{,}06d_3 + 8{,}72d_4 + ... + 47{,}88d_{12}$$

El MAE en el conjunto de *test* es de 19.44597. Seguimos teniendo residuos con correlaciones significativas.



Triple suavizado exponencial

Este modelo realiza un suavizado exponencial a varias componentes para después combinarlas y realizar las predicciones.

A continuación se muestra el caso aditivo:

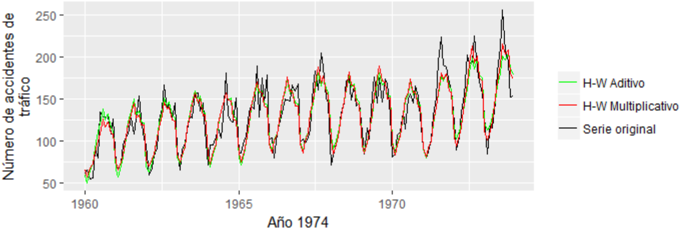
$$
\begin{aligned}
\widehat{Y}_{t+h} &= l_t + hb_t + S_{t-M+h_M^+} \\
l_t &= \alpha(Y_t - S_{t-M}) + (1-\alpha)(l_{t-1} + b_{t-1}) \\
b_t &= \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \\
S_t &= \gamma(Y_t - l_{t-1} + b_{t-1}) + (1-\gamma)S_{t-M}
\end{aligned}
$$

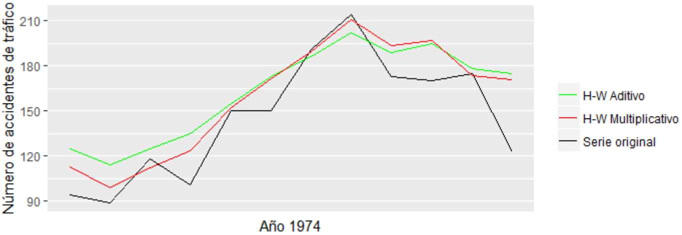
Los coeficientes α, β y γ determinan la importancia de las observaciones pasadas en la predicción.

Es posible añadir un parámetro ϕ de suavización o amortiguación de la tendencia (*damped trend*).

Para ajustar este tipo de modelos recurrimos al método `hw()`.

```
hw(y, h = 2*frequency(x), seasonal = c("additive", "multiplicative"),
   damped = FALSE, initial = c("optimal", "simple"), alpha = NULL,
   beta =  NULL, gamma = NULL, phi = NULL, ...)
```





El MAE en estos modelos es de 19.67722 bajo estructura aditiva y 15.04762 bajo estructura multiplicativa.

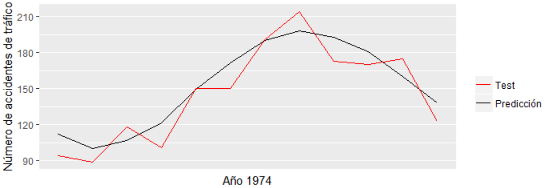
Red neuronal

Con el método `nnetar()` es posible implementar una red NNAR(p,P,k)$_M$ siendo:

- p los rezagos que toma como *inputs*.
- P los periodos anteriores de los que toma los valores correspondientes como *inputs*.
- k el número de neuronas de la capa intermedia.

$$Y_{t-1}, Y_{t-2}, ..., Y_{t-p}, Y_{t-M}, Y_{t-2M}, ..., Y_{t-PM}$$

En este tipo de modelos el parámetro de regularización juega un papel fundamental ya que nos ayuda a controlar el *overfitting*.



Obtenemos un MAE de 13.29118.

Modelos ARIMA

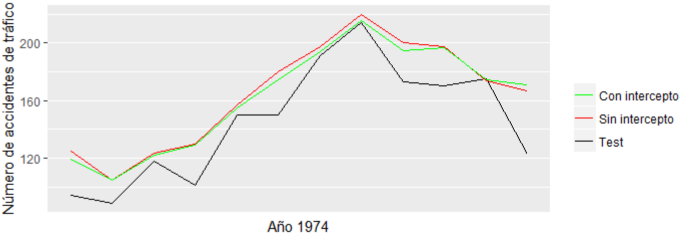
Esta librería implementa estos modelos a través de `Arima()`.

```
Arima(y, order = c(0,0,0), seasonal = c(0,0,0),
      include.mean = TRUE, include.drift = FALSE, ...)
```

Para una selección óptima del modelo nos podemos apoyar en:

- Análisis de la FAC y FACP a través de `Acf()` y `Pacf()`.
- Determinar diferencias a partir de `ndiffs()` y `nsdiffs()`.

Hemos optado por ajustar un $\text{SARIMA}(1,0,1)(2,1,1)_{12}$.



Es posible automatizar todo este proceso a través de `auto.arima()`.

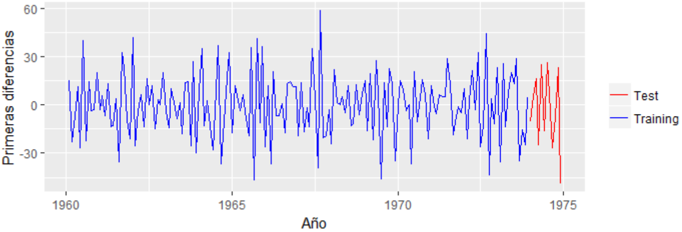
```
auto.arima(y, d = NA, D = NA, max.p = 5, max.q = 5, max.P = 2,
          max.Q = 2, max.d = 2, max.D = 1, ic = ("aicc", "aic", "bic"),
          test = c("kpss", "adf", "pp"),
          seasonal.test = c("ocsb", "ch"),...)
```

A través de sus argumentos es posible personalizar la búsqueda.

	AIC	AICc	BIC	MAE	Ljung Box
$(0,1,0)(0,0,2)_{12}$	1550.38	1550.53	1559.73	22.19949	~ 0
$(1,0,1)(2,1,1)_{12\ \textit{sin int}}$	1367.68	1368.25	1385.98	19.22651	0.04321
$(1,0,1)(2,1,1)_{12\ \textit{con int}}$	**1359.82**	**1360.58**	**1381.17**	17.09992	**0.2748**
$(1,0,0)(1,0,0)_{12}$	1514.22	1514.47	1526.72	**14.86938**	0.00085
$(2,0,0)(2,0,0)_{12}$	1499.24	1499.77	1517.99	**14.96979**	0.002708
$(1,1,4)(2,0,0)_{12}$	1485.29	1486.2	1510.24	23.44705	0.01353

TSeries Package

Necesitamos diferenciar y desestacionalizar la serie para trabajar con esta librería.



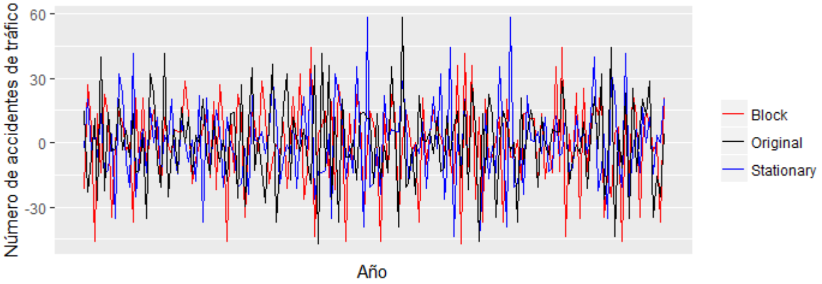
Parte del preprocesado se centra en la implementación de tests estadísticos:

- Test aumentado de Dickey-Fuller (`adf.test()`)
- Test de Philipps-Perron (`pp.test()`)
- Test de KPSS (`kpss.test()`)
- ...

Técnicas de remuestreo

Hay dos formas de implementar estas técnicas con esta librería:

- `tsbootstrap()`: implementa un *bootstrap* por bloques. Existen dos variaciones:
  - **Por bloques**: La serie se divide en bloques de longitud b. Se hace un remuestreo aleatorio con reposición y se combinan respetando el orden temporal original.
  - **Estacionario**: La longitud b varía aleatoriamente.
- `surrogate()`: Recurre a transformadas de Fourier para generar remuestreos de la serie original.

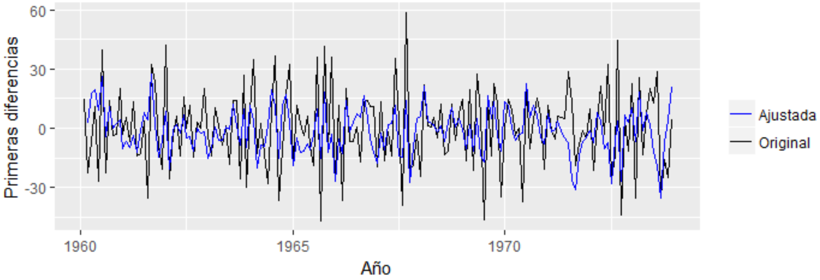


Modelos ARMA

Estos modelos se implementan a través de `arma()`.

```
arma(x, order = c(1, 1), lag = NULL, coef = NULL,
     include.intercept = TRUE, qr.tol = 1e-07, ...)
```

Hemos ajustado un ARMA$(2,1)$.



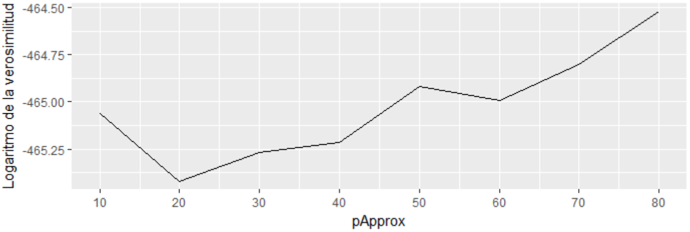
Esta librería incluye métodos como `jarque.bera.test()` y `bds.test()` para estudiar lo residuos a través de tests estadísticos.

FitARMA Package

Esta librería está enfocada a implementar modelos ARIMA.
Se caracteriza por utilizar un método de estimación que optimiza el ajuste del modelo.
Se implementa a través de `FitARMA()`.

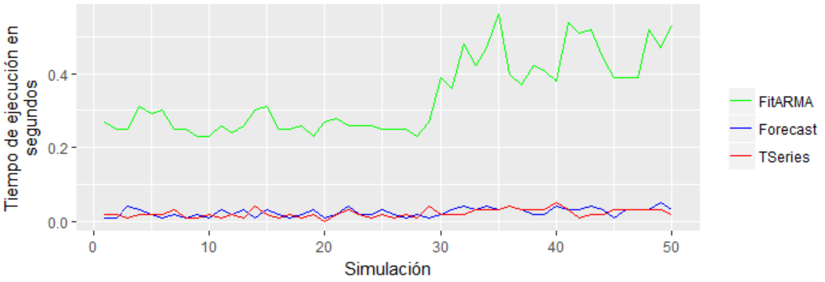
```
FitARMA(z, order = c(0, 0, 0), demean = TRUE, MeanMLEQ = FALSE,
        pApprox = 30, MaxLag = 30)
```

El parámetro `pApprox` juega un papel fundamental en el ajuste.



Se ha llevado a cabo una simulación para estudiar el tiempo de ejecución de los métodos encargados de implementar los modelos ARMA/ARIMA de las tres librerías:

- `Arima()` de `forecast`
- `arma()` de `TSeries`
- `FitARMA()` de `FitARMA`



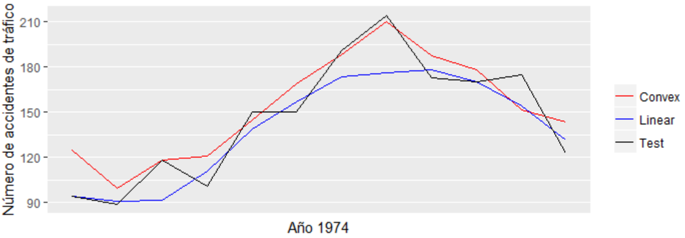
Opera Package

Esta librería nos permite combinar modelos a fin de obtener mejores predicciones.

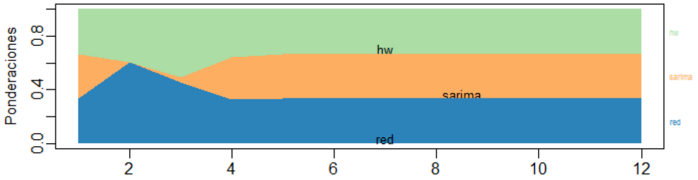
$$\widehat{Y}_{t+h} = \sum_{k=1}^{K} p_{k,t+h} \; x_{k,t+h}$$

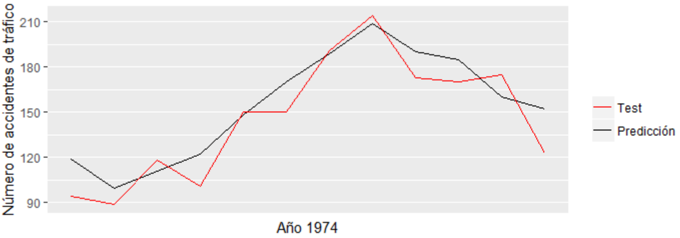
Hay dos formas de implementar esto:

- **Combinaciones fuera de línea**: Se implementa a través de `oracle()`. Ajusta las ponderaciones de acuerdo al poder predictivo de cada modelo. Es posible combinar las predicciones de dos formas:
  - Combinación lineal
  - Combinación convexa

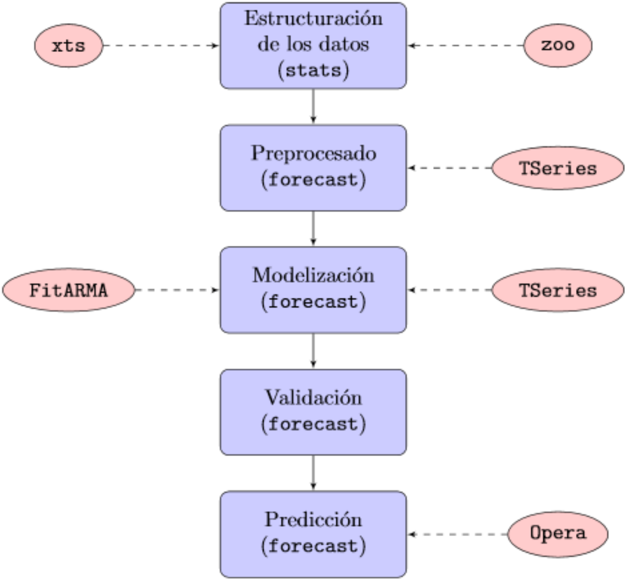


- **Combinaciones en línea**: Implementa *aggregation rules* a través de `mixture()`. Son útiles para ajustar las ponderaciones secuencialmente.





Conclusiones



Pero... ¿ha merecido la pena?


SERIE TEMPORAL
¿Qué es una serie temporal?
Secuencia de variables aleatorias $Y_1, Y_2, Y_3 \ldots \rightarrow \{Y_t\}$ con $t \in N$
Estas variables están correlacionadas temporalmente
Parte determinista y parte aleatoria

Gracias por su atención