Series de tiempo

Paula Vargas Pellicer

28/04/2022

Series de tiempo

En esta clase, vas a explorar y analizar datos de series de tiempo; es una técnica poderosa que se puede usar para comprender los diversos patrones temporales en nuestros datos al descomponerlos en diferentes tendencias cíclicas. El análisis de series de tiempo también se puede usar para predecir cómo cambiarán los niveles de una variable en el futuro, teniendo en cuenta lo que sucedió en el pasado.

Carga los paquetes necesarios para esta clase. No olvides instalar primero los paquetes que aún no tengas.

Carga las bases de datos.

```
library(ggplot2)
library(forecast)

library(dplyr)

library(colortools)

monthly_milk <- read.csv("~/Downloads/CC-time-series-master/monthly_milk.csv"
)  # Producción de leche por vaca por mes
daily_milk <- read.csv("~/Downloads/CC-time-series-master/daily_milk.csv")  #
Producción de leche por vaca por ordeñada
```

1. Formatear datos de series temporales

El primer problema cuando se usan datos de series temporales es ponerlos en un formato que sea fácil de leer para R. Un formato común para los datos de series de tiempo pone la porción de tiempo más grande primero (por ejemplo, año) y se vuelve progresivamente más pequeña, así: `2017-02-25 18:30:45`

Esto se puede generalizar a AAAA-MM-DD HH:MM:SS. (Si el tiempo no es importante para tus medidas, puedes quitar HH:MM:SS de los registros).

Los datos en monthly_milk muestran la producción mensual de leche de un rebaño de ganado entre 1962 y 1975. Primero, debemos revisar la base de datos:

```
head(monthly_milk)

class(monthly_milk)
```

```r
class(monthly_milk$month)
```

Si tratáramos de analizar estos datos en su forma actual, R no entendería que 1962-01-01 representa un punto en el tiempo y viene un mes antes de 1962-02-01. Afortunadamente, R también tiene la clase Fecha (Date), con la que es mucho más fácil trabajar. Así que coaccionemos los datos a la clase Date:

```r
monthly_milk$month_date <- as.Date(monthly_milk$month, format = "%Y-%m-%d")
class(monthly_milk$month_date)
```

Aquí hay un ejemplo de los códigos de el formato de fecha:

Name	Code	Example
Long year	%Y	2017
Short year	%y	17
Numeric month	%m	02
Abbreviated month	%b	Feb
Full month	%B	February
Day of the month	%d	25
Abbreviated weekday	%a	Sat
Full weekday	%A	Monday
Day of the week (1-7)	%u	6
Day of the year	%j	56

Para transformar un objeto de clase Fecha en un formato de carácter puedes usar también format() junto con cualquiera de los códigos de fecha de la tabla anterior. Por ejemplo, podríamos transformar 2017-02-25 en febrero - sábado 25 - 2017. Pero ten en cuenta que R no interpretará este nuevo formato de carácter como una fecha en los análisis.


Prueba algunas combinaciones diferentes de códigos de fecha de la tabla anterior, utilizando el siguiente código como ejemplo (ojo, no se asignarán los resultados a un objeto, sino que simplemente los imprimirá en la consola):

```r
format(monthly_milk$month_date, format = "%Y-%B-%u")

class(format(monthly_milk$month_date, format = "%Y-%B-%u"))
```

Fechas y horas

A veces, tanto la fecha como la hora de la observación son importantes. La mejor manera de formatear la información de la hora es agregarla después de la fecha en la misma columna de esta manera: `2017-02-25 18:30:45` La clase más apropiada y utilizable para estos datos es POSIXct POSIXt. Para explorar esta clase de datos, veamos otro conjunto de datos de producción de leche, esta vez con mayor resolución, que muestra los tiempos de ordeño matutino y vespertino en el transcurso de algunos meses:

```
head(daily_milk)

class(daily_milk$date_time)
```

Nuevamente, la fecha y la hora están en un formato razonable (AAAA-MM-DD HH:MM:SS), pero R las interpreta como parte de la clase de caracteres. Cambiemos esto a POSIXct POSIXt usando as.POSIXct():

```
daily_milk$date_time_posix <- as.POSIXct(daily_milk$date_time, format = "%Y-%
m-%d %H:%M:%S")
class(daily_milk$date_time_posix)
```

Corrección de datos de fecha mal formateados

Si tus datos aún no están bien formateados es fácil volver a transformarlos en un formato utilizable usando format(), antes de transformarlos a la clase Fecha. Primero, vamos a crear algunos datos de fecha mal formateados para que se parezcan a 01/dic/1975-1 (el día del mes, el mes abreviado, el año y el día de la semana):

```
monthly_milk$bad_date <- format(monthly_milk$month_date, format = "%d/%b/%Y-%
u")
head(monthly_milk$bad_date)

class(monthly_milk$bad_date)
```

Luego, para volver a transformarlo al útil formato de fecha AAAA-MM-DD, simplemente usa as.Date(), especificando el formato en el que se encuentran los datos mal formateados:

```
monthly_milk$good_date <- as.Date(monthly_milk$bad_date, format = "%d/%b/%Y-%
u")
head(monthly_milk$good_date)

class(monthly_milk$good_date)
```

2. Visualización de datos de series temporales

Trazar datos de series de tiempo con ggplot2 requiere el uso de scale_x_date() para construir correctamente las etiquetas de los ejes y permitir una fácil personalización de las marcas de los ejes:

```
(time_plot <- ggplot(monthly_milk, aes(x = month_date, y = milk_prod_per_cow_
kg)) +
```

```
    geom_line() +
    scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
    theme_classic())
```

El uso de theme_classic() produce una grafica que es un poco más agradable estéticamente que las opciones predeterminadas.

☛ Juega con date_labels, reemplazando "%Y" con algunas otras marcas de fecha de la tabla anterior (por ejemplo, %m-%Y). date_breaks también se puede personalizar para cambiar la frecuencia de la etiqueta del eje. Otras opciones incluyen mes, semana y día (por ejemplo, "5 semanas")

El trazado de datos de fecha y hora se realiza de manera similar utilizando scale_x_datetime():

```
(time_plot_2 <- ggplot(daily_milk, aes(x = date_time_posix, y = milk_prod_per
_cow_kg)) +
    geom_line() +
    scale_x_datetime(date_labels = "%p-%d", date_breaks = "36 hour") +
    theme_classic())
```

3. Análisis estadístico de datos de series temporales

Descomposición

Los datos de series temporales pueden contener múltiples patrones que actúan en diferentes escalas temporales. El proceso de aislar cada uno de estos patrones se conoce como descomposición. Echa un vistazo a un gráfico simple de `monthly_milk` como el que vimos antes:

```
(decomp_1 <- ggplot(monthly_milk, aes(x = month_date, y = milk_prod_per_cow_k
g)) +
    geom_line() +
    scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
    theme_classic())
```

En primer lugar, parece que hay una tendencia general al alza: se produce más leche en 1975 que en 1962. Esto se conoce como un patrón "suave", que aumenta o disminuye regularmente (monótonamente) a lo largo de la serie temporal. Podemos ver este patrón más claramente trazando una regresión de loess (una regresión polinominal) a través de los datos. Una regresión de loess ajusta una curva suave entre dos variables.

```
(decomp_2 <- ggplot(monthly_milk, aes(x = month_date, y = milk_prod_per_cow_k
g)) +
    geom_line() +
    geom_smooth(method = "loess", se = FALSE, span = 0.6) +
    theme_classic())
```

El término *span* establece el número de puntos que se utilizan para trazar cada regresión local en la curva: cuanto menor sea el número, más puntos se utilizarán y más se ajustará la curva a los datos originales.

Parece que hay algunos picos y valles que ocurren regularmente cada año. Este es un patrón "estacional". Podemos investigar más este patrón trazando cada año como su propia línea y comparando los diferentes años:

```
# Extrae mes y año en columnas separadas

monthly_milk$year <- format(monthly_milk$month_date, format = "%Y")
monthly_milk$month_num <- format(monthly_milk$month_date, format = "%m")

# Crea una paleta de colores usando `colortools`
year_pal <- sequential(color = "darkturquoise", percentage = 5, what = "value
")

# Haz la grafica
(seasonal <- ggplot(monthly_milk, aes(x = month_num, y = milk_prod_per_cow_kg
, group = year)) +
    geom_line(aes(colour = year)) +
    theme_classic() +
    scale_color_manual(values = year_pal))
```

De la gráfica se desprende claramente que, si bien la producción de leche aumenta constantemente, el mismo patrón ocurre a lo largo de cada año, con un pico en mayo y un mínimo en noviembre.

Las tendencias “cíclicas” son similares a las tendencias estacionales en que se repiten con el tiempo, pero ocurren en escalas de tiempo más largas. Puede ser que la tendencia ascendente general y la meseta que se observan con la regresión de loess sean parte de un ciclo decenal más largo, pero esto es imposible de probar sin una serie de tiempo más larga.

Un método alternativo para generar estos gráficos en ggplot2 es convertir la base de datos de la serie temporal en un objeto de clase `ts` y descomponerlo usando `stl()` del paquete stats. Esto reduce la capacidad de personalizar los gráficos, pero es más rápido:

```
# Transforma a clase `ts`
monthly_milk_ts <- ts(monthly_milk$milk_prod, start = 1962, end = 1975, freq
= 12)

# Decomponer usando `stl()`
monthly_milk_stl <- stl(monthly_milk_ts, s.window = "period")

plot(monthly_milk_stl)

monthplot(monthly_milk_ts)  # variación de producción de leche por cada mes

seasonplot(monthly_milk_ts)
```

Pronóstico

A menudo, los datos de series temporales se utilizan para predecir lo que podría suceder en el futuro, dados los patrones observados en los datos. Esto se conoce como pronósitco. Hay muchos métodos que se usan para pronosticar datos de series de tiempo, y varían ampliamente en complejidad, pero esto debería servir como una breve introducción a los métodos más comúnmente usados.

Todos los modelos utilizados hoy se conocen como modelos ETS. ETS significa Error, Tendencia, Estacionalidad. Los modelos ETS también se conocen como modelos de espacio de estado de suavizado exponencial. Los modelos ETS se utilizan para modelar cómo una sola variable cambiará con el tiempo mediante la identificación de sus tendencias subyacentes, sin tener en cuenta ninguna otra variable. Los modelos ETS se diferencian de un promedio móvil simple al ponderar la influencia de los puntos anteriores en los puntos de tiempo futuros en función de cuánto tiempo hay entre los dos puntos. Es decir, durante un período de tiempo más largo es más probable que alguna condición no medida haya cambiado, dando como resultado un comportamiento diferente de la variable que se ha medido. Otro grupo importante de modelos de pronóstico son los modelos ARIMA, modelos autorregresivos que describen autocorrelaciones en los datos en lugar de tendencias y estacionalidad.

Los modelos ETS normalmente se indican con tres letras, p. ETS_AMZ. La primera letra (A) se refiere al tipo de error, la segunda letra (M) es el tipo de tendencia y la tercera letra (Z) es el tipo de temporada. Las letras posibles son: N = Ninguno

A = Aditivo

M = Multiplicativo

Z = Seleccionado automáticamente

No nos vamos a preocupar demasiado por las implicaciones de estos tipos de modelos por ahora. Para esta clase, solo elegiremos algunos tipos de modelos básicos y los compararemos.

Elegir qué modelo usar para pronosticar datos puede ser difícil y requiere usar un poco de intuición, así como hacer estadísticas de prueba. Para probar la precisión de un modelo, tenemos que compararlo con datos que no se han usado para generar el pronóstico, así que vamos a crear algunos subconjuntos de datos del objeto de serie de tiempo month_milk_ts: uno para generar el modelo (monthly_milk_model) y otro para probar la precisión del modelo (monthly_milk_test). window() es una función similar a subset() o filter(), que subdivide un objeto en función de argumentos, pero se usa especialmente para objetos de series temporales (ts). window() toma el objeto de la serie temporal original (x) y los puntos de inicio y finalización del subconjunto. Si no se incluye el final, el subconjunto se extiende hasta el final de la serie temporal:

```
monthly_milk_model <- window(x = monthly_milk_ts, start = c(1962), end = c(19
70))
monthly_milk_test <- window(x = monthly_milk_ts, start = c(1970))
```

Primero comparemos los pronósticos de modelos de diferentes modelos ETS visualmente extrayendo datos de pronóstico de objetos de pronóstico y trazándolos usando ggplot() contra los datos de prueba. El siguiente código es bastante largo y podría hacerse más conciso mediante el uso de canalizaciones o funciones apply(), pero escribirlo a mano de esta manera permite investigar todos los objetos intermedios por separado para comprender cómo están estructurados y qué muestran:

```
# Crea objetos de modelos de cada tipo de modelo ets
milk_ets_auto <- ets(monthly_milk_model)
milk_ets_mmm <- ets(monthly_milk_model, model = "MMM")
milk_ets_zzz<- ets(monthly_milk_model, model = "ZZZ")
milk_ets_mmm_damped <- ets(monthly_milk_model, model = "MMM", damped = TRUE)

# Crea objecto pronóstico de cada objeto de modelo
milk_ets_fc <- forecast(milk_ets_auto, h = 60)  # `h = 60` el pronostico va a
durar 60 dias (un mes)
milk_ets_mmm_fc <- forecast(milk_ets_mmm, h = 60)
milk_ets_zzz_fc <- forecast(milk_ets_zzz, h = 60)
milk_ets_mmm_damped_fc <- forecast(milk_ets_mmm_damped, h = 60)

# Convierte el pronóstico en base de datos
milk_ets_fc_df <- cbind("Month" = rownames(as.data.frame(milk_ets_fc)), as.da
ta.frame(milk_ets_fc))
names(milk_ets_fc_df) <- gsub(" ", "_", names(milk_ets_fc_df))  # Quita espac
ios en blanco de los nombres de las columnas
milk_ets_fc_df$Date <- as.Date(paste("01-", milk_ets_fc_df$Month, sep = ""),
format = "%d-%b %Y")  # antepón el día del mes
milk_ets_fc_df$Model <- rep("ets")  # Agrega columna del tipo de modelo

milk_ets_mmm_fc_df <- cbind("Month" = rownames(as.data.frame(milk_ets_mmm_fc)
), as.data.frame(milk_ets_mmm_fc))
names(milk_ets_mmm_fc_df) <- gsub(" ", "_", names(milk_ets_mmm_fc_df))
milk_ets_mmm_fc_df$Date <- as.Date(paste("01-", milk_ets_mmm_fc_df$Month, sep
= ""), format = "%d-%b %Y")
milk_ets_mmm_fc_df$Model <- rep("ets_mmm")

milk_ets_zzz_fc_df <- cbind("Month" = rownames(as.data.frame(milk_ets_zzz_fc)
), as.data.frame(milk_ets_zzz_fc))
names(milk_ets_zzz_fc_df) <- gsub(" ", "_", names(milk_ets_zzz_fc_df))
milk_ets_zzz_fc_df$Date <- as.Date(paste("01-", milk_ets_zzz_fc_df$Month, sep
= ""), format = "%d-%b %Y")
milk_ets_zzz_fc_df$Model <- rep("ets_zzz")

milk_ets_mmm_damped_fc_df <- cbind("Month" = rownames(as.data.frame(milk_ets_
mmm_damped_fc)), as.data.frame(milk_ets_mmm_damped_fc))
names(milk_ets_mmm_damped_fc_df) <- gsub(" ", "_", names(milk_ets_mmm_damped_
fc_df))
milk_ets_mmm_damped_fc_df$Date <- as.Date(paste("01-", milk_ets_mmm_damped_fc
_df$Month, sep = ""), format = "%d-%b %Y")
```

```
milk_ets_mmm_damped_fc_df$Model <- rep("ets_mmm_damped")

# Combina en una base de datos
forecast_all <- rbind(milk_ets_fc_df, milk_ets_mmm_fc_df, milk_ets_zzz_fc_df,
milk_ets_mmm_damped_fc_df)
```

Ahora si, una grafica

```
(forecast_plot <- ggplot() +
    geom_line(data = monthly_milk, aes(x = month_date, y = milk_prod_per_cow_
kg)) +  #la información original
    geom_line(data = forecast_all, aes(x = Date, y = Point_Forecast, colour =
Model)) +  # el pronóstico
    theme_classic())
```

También se puede comparar numéricamente la precisión de diferentes modelos con los datos que excluimos del modelo (monthly_milk_test) usando accuracy():

```
accuracy(milk_ets_fc, monthly_milk_test)

accuracy(milk_ets_mmm_fc, monthly_milk_test)

accuracy(milk_ets_zzz_fc, monthly_milk_test)

accuracy(milk_ets_mmm_damped_fc, monthly_milk_test)
```

Vamos a separar esas estadísticas:

ME: Error medio: la diferencia media entre los valores modelados y observados

RMSE: Error cuadrático medio de la raíz. Toma cada diferencia entre el modelo y los valores observados, eleva al cuadrado, toma la media y luego hace la raíz cuadrada.

MAE: Error absoluto medio. Lo mismo que ME, pero todos los errores se transforman en valores positivos para que los errores positivos y negativos no se anulen entre sí.

MPE: Error porcentual medio. Similar a ME, pero cada error se expresa como un porcentaje de la estimación del pronóstico. Los errores porcentuales no dependen de la escala, por lo que se pueden usar para comparar la precisión del pronóstico entre conjuntos de datos.

MAPE: Error porcentual absoluto medio. Lo mismo que MPE, pero todos los errores se transforman en valores positivos para que los errores positivos y negativos no se anulen entre sí.

MASE: Error escalado absoluto medio. Compara el MAE del pronóstico con el MAE producido por un pronóstico ingenuo. Un pronóstico ingenuo es aquel que simplemente proyecta una línea recta hacia el futuro, cuyo valor es el valor final de la serie temporal utilizada para construir el modelo. Un MASE>1 nos dice que el pronóstico ingenuo se ajusta mejor a los datos observados que el modelo, mientras que un MASE<1 nos dice que el modelo fue mejor que el modelo ingenuo.

ACF1: función de correlación automática en el retraso 1. ¿Qué tan correlacionados están los puntos de datos con los puntos de datos directamente después de ellos, donde ACF = 1 significa que los puntos están completamente correlacionados y ACF = 0 significa que los puntos no están en absoluto correlacionados?

U de Theil: compara el pronóstico con los resultados de un modelo utilizando datos mínimos. Los errores se elevan al cuadrado para dar más peso a los errores grandes. U<1 significa que el pronóstico es mejor que adivinar, mientras que U>1 significa que el pronóstico es peor que adivinar.

MAPE es la medida de precisión de pronóstico más utilizada, probablemente debido a que es fácil de entender conceptualmente. Sin embargo, MAPE se vuelve muy sesgado cuando los valores observados en la serie de tiempo son cercanos a cero e infinito cuando las observaciones son iguales a cero, lo que lo hace inadecuado para algunas series de tiempo que tienen valores de informe bajos. MAPE también otorga una penalización mayor a las desviaciones positivas que a las negativas, lo que lo hace útil para algunos análisis, p..

Al comparar las estadísticas MAPE y MASE de los cuatro modelos en la fila Conjunto de prueba, podemos ver que los modelos month_milk_ets_fc y month_milk_ets_zzz_fc tienen los valores más bajos. Al observar los gráficos de este pronóstico y compararlo visualmente con los datos de prueba, podemos ver que este es el pronóstico que mejor se ajusta a los datos de prueba. Entonces podemos usar ese pronóstico para proyectar hacia el futuro.

Extraer valores de un pronóstico

Ahora que hemos identificado los mejores modelos de pronóstico, podemos usar estos modelos para averiguar cómo será la producción de leche en el año 1975. Utiliza el siguiente código para extraer un valor predicho para un año determinado de nuestros pronósticos. Esto es tan simple como subdividir la base de datos de pronóstico para extraer el valor correcto.

```
milk_ets_fc_df %>%
    filter(Month == "Jan 1975") %>%
    select(Month, Point_Forecast)

milk_ets_zzz_fc_df %>%
    filter(Month == "Jan 1975") %>%
    select(Month, Point_Forecast)
```

4. Desafío de codificación

Ahora usa lo que ha aprendido para hacer algunos modelos pronóstico y trazar algunas gráficas para investigar patrones temporales para nuestros datos sobre concentraciones de CO^2 en Mauna Loa, Hawái. Ve si puedes predecir la concentración de CO^2 para junio de 2050. Puedes encontrar los datos en *co2_loa.csv* en la carpeta del repositorio de GitHub para esta clase.