Series de tiempo en R (Documento 1)

Introducción a las Series de tiempo

Jose Oscar Henao Monje

Bienvenida

Hola a todos y todas, mi nombre es Jose Oscar Henao Monje[1] y les acompañaré en un interorización a las series de tiempo en el software RStudio. Durante esta sesión aprenderemos cómo manerar el tiempo y formatos fechas.Muchos éxitos!!!

Primero que todo limpiamos nuestra mesa de trabajo:

```
rm(list = ls()) #Limpiar el ambiente
cat("\f") #Limpiar la consola
```

[1]Economista colombiano y candidato a Magister en Economía de la Universidad de Buenos Aires. Es investigador principal del Programa Internacional sobre Democracia, Sociedad y Nuevas Economías - PIDESONE de la Universidad de Buenos Aires. Coordina actualmente el Observatorio Regional para el Desarrollo Sostenible de Cáritas en América Latina y el Caribe y es consultor de organizaciones internacionales como el Programa de Naciones Unidas para el Desarrollo, Banco Interamericano de Desarrollo. Sus temas de interés: Series de tiempo, microeconometría, evaluación de impacto, desarrollo sostenible, pobreza y desigualdad y mercado de trabajo.

```
dev.off() #Limpiar plots
```

```
## null device
##           1
```

Introducción a las librerías

Como bien saben este programa cuenta con diferentes funcionalidades agrupadas a través de paquetes que facilitan el análisis de diferentes tópicos. Para el caso de Series de Tiempo (ST) utilizaremos 3 paquetes iniciales.

1. Paquete "lubridate"
2. Paquete "tseries"
3. Paquete "forecast"

Pero antes de instalar les sugiero ingresen a la solapa "Packages" de la parte inferior derecha.

Una vez verificado en "Packages" que los 3 paquetes que queremos utilizar existen o no, usted debe instalar el paquete respectivo (en caso de no existir), o simplemente cargar la librería correspondiente.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(forecast)
```

- lubridate: no es un paquete especifico, es un paquete para pre-procesado de dato y se utiliza para formatear de la variable donde tenemos la fecha, el tiempo.
- tseries: nos permite trabajar con el objeto serie temporal (modelar, graficar, testear)
- forecast: Tiene muchas funciones de pronosticos, tiene herramientas para limpiar datos, outliers.Herramientas gráficas de series de tiempo o pronósticos futuros.

Formato de datos a tiempo

Trabajaremos diferentes formas de darle formato al tiempo.

Forma 1

Este primer procedimiento permite que una variable sea serie de tiempo (tipo fecha o tiempo). Las opciones que tenemos disponibles en R base son la clase "POSIXt" que significa en inglés "Portable Operating System Interface for Time" (interface de sistema operativo para el tiempo), que es la codificación extandar para el tiempo.

Vamos a guardar en x el resultado de convertir un solo periodo de tiempo en la clase POSIXt que tiene dos variantes (ct y lt):

```
x <- as.POSIXct("2021-08-12 20:00:34") # Números de segundos
y <- as.POSIXlt("2021-08-12 20:00:34")
```

Ahora ¿Qué implica que hallamos incorporado estas dos variantes de la clase POSIXt? verificando qué clase es x e y, tenemos que:

```
unclass(x)
```

```
## [1] 1628809234
## attr(,"tzone")
## [1] ""
```

```
unclass(y)
```

```
## $sec
## [1] 34
##
## $min
## [1] 0
##
## $hour
## [1] 20
##
## $mday
## [1] 12
##
## $mon
## [1] 7
##
## $year
## [1] 121
##
## $wday
## [1] 4
##
## $yday
## [1] 223
##
## $isdst
## [1] 0
##
## $zone
## [1] "-03"
```

```
##
## $gmtoff
## [1] NA
```

Nota: La función unclass () permite visualizar cómo R almacena la información de la variable x tipo factor.

"La clase POSIXct almacena internamente esta cifra como un número entero, mientras que la clase POSIXlt la descompone en una lista en diferentes elementos para los segundos, minutos, horas, día, mes y año."[2]

¿Qué significa este número 1628779534? Corresponde a la cantidad de segundo que hay hasta esa fecha, desde un punto de referencia 01-01-1970 00:00:00. Este punto de referencia es el que toma la función POSIXct, sin contar años bisiestos.

Forma 2

Otra forma de trabajar series y tiempos es a través de "as.Date". Aquí utilizamos as.punto algo quiere decir que queremos convertir esto en algo bajo un formato específico.

```
x = as.Date("2021-08-12")
```

¿Qué clase tendría este "as.Date"? Verifiquemos!!!

```
x; class(x)
```

```
## [1] "2021-08-12"
```

```
## [1] "Date"
```

```
unclass(x)
```

```
## [1] 18851
```

Y efectivamente nos confirma que está en formato fecha y nos menciona efectivamente que corresponde a 12 de agosto de 2021. Y el unclass nos devuelve el número de días desde el punto de referencia mencionado.

Forma 3

Una tercera forma de dar formato es a través del paquete "chron" por lo que nuevamente les invito a verificar si lo tienen instalado, de lo contrario instálelo a través del siguiente comando: > install.packages("chron"). De tenerlo instalado cargue la librería respectiva.

```
library(chron)
```

```
##
## Attaching package: 'chron'
```

```
## The following object is masked from 'package:tseries':
##
##     is.weekend
```

[2]Tomado de: https://estadistica-dma.ulpgc.es/cursoR4ULPGC/6h-Fechas.html

```
## The following objects are masked from 'package:lubridate':
##
##     days, hours, minutes, seconds, years
```

Ahora podemos crear nuevamente un valor x que contiene la fecha a través del comando chron.Verifiquemos 3 cosas: 1) qué valor tiene x, 2) qué clase es x y 3) identifique cuantos días han pasado desde el punto de referencia (unclass de x).

```
x = chron("12/08/2021", "20:00:34")
x
```

```
## [1] (12/08/21 20:00:34)
```

```
class(x)
```

```
## [1] "chron" "dates" "times"
```

```
unclass(x)
```

```
## [1] 18969.83
## attr(,"format")
##   dates   times
## "m/d/y" "h:m:s"
## attr(,"origin")
## month   day  year
##     1     1  1970
```

El paquete "chron" permite transformar objetos a diferentes formatos temporales (fecha, hora,)

Forma 4

Utilizar la función strptime que convierte caracteres de texto a tiempo (strings to data and time). Supongamos que tenemos un valor a que contiene un vector de 3 momentos en el tiempo y los convierte en caracter.

```
a = as.character(c("2021-08-12 20:00", "2021-09-25 12:01", "2021-03-09 11:46"))
class(a)
```

```
## [1] "character"
```

Para darle formato de tiempo, utilizamos la función strptime y la ubicaremos en b:

```
b = strptime(a, format ="%Y-%m-%d %H:%M") # format en la función permite
#decirle de qué forma queremos que se interprete el texto
b #visualicemos b
```

```
## [1] "2021-08-12 20:00:00 -03" "2021-09-25 12:01:00 -03"
## [3] "2021-03-09 11:46:00 -03"
```

```
class(b) #verifiquemos que clase es.
```

```
## [1] "POSIXlt" "POSIXt"
```

Quieres profundizar otros tipos de formato que se le puede asignar no dudes en utilizar el comando: ">?strptime"

Forma 5

Utilizar el paquete lubridate, este es el paquete más utilizado, nos genera muchos beneficios para manejar tiempos y fechas en diferentes formas.Simplemente ponemos el orden temporal que quisieramos tener. Por ejemplo para el 12 de agosto de 2021 y queremos formato Año/Mes/Día el número a incorporar en la función ymd es 20210812.

```
ymd(20210812) # Año / Mes / Día
```

```
## [1] "2021-08-12"
```

```
dmy(12082021)
```

```
## [1] "2021-08-12"
```

```
mdy(08122018)
```

```
## [1] "2018-08-12"
```

Este reconoce inmediatamente los valores incorporados como fecha y toma el formato elegido.

También podemos utilizar fecha y tiempo a la vez: Ejemplo la fecha de nuestra primera clase de series de tiempo y la vamos a incorporar en la variable miprimeraclase.

```
primeraclase <- ymd_hm("2021-08-12 20:00")
primeraclase
```

```
## [1] "2021-08-12 20:00:00 UTC"
```

```
class(primeraclase)
```

```
## [1] "POSIXct" "POSIXt"
```

Objeto serie temporal e introducción a los gráficos

Antes de iniciar con esta sección quisiera invitarles a visitar algunas herramientas diseñadas para fortalecer su aprendizaje en RStudio respecto al abordaje de las series de tiempo. En la parte inferior izquierda podrán encontar un simbolo de una casita, al hacer clic allí se podrá encontrar diferentes recursos de consulta, uno de los que más utilizo corresponde a los "RStudio Cheat Sheets". Puede ingresar también a través del siguiente link: https://www.rstudio.com/resources/cheatsheets/

Para practicar alguna programación en relación a diferentes paquetes de R para ST, he preparado la implementación del paquete "tsbox" que tiene como objeto pasar objetos de series temporales de un formato a otro de manera sencilla. Para conocer un poco de este les invito a visitar el siguiente link: (Insertar link de medium.com/johenaom).

Para trabajar un objeto de serie de tiempo tienes diferentes formas por ahora trabajaremos con la creación de unos datos.

Creación de unos datos

Se simula unos datos uniformemente distribuidos, un total de 50 datos para empezar con valores entre 10 y 45. Esto a través de la función runif.

```
misdatos = runif(n=50, min=10, max=45)
class(misdatos)
```

```
## [1] "numeric"
```

Una vez creado estos datos, estamos en condiciones de convertir mis datos en una serie de tiempo a través de la función ts.

Creación de serie de tiempo

Creamos una serie de tiempo a partir de los datos creados en el objeto misdatos, especificamos que sea a partir de 1956 con una frecuencia trimestral.

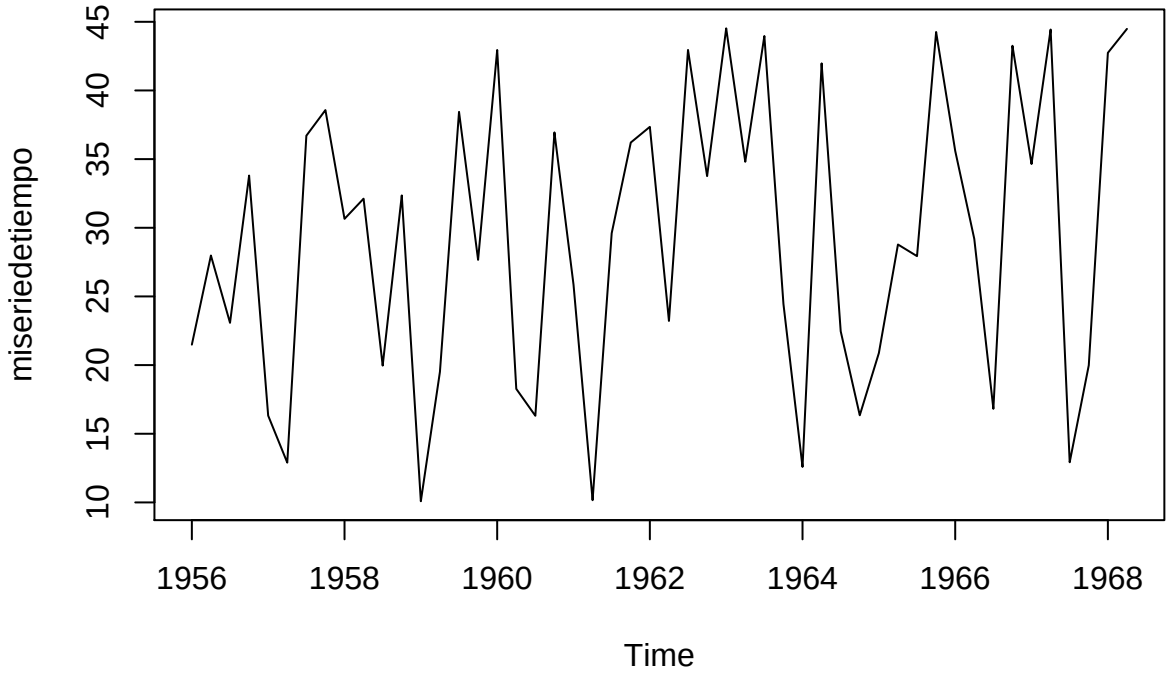
```
miseriedetiempo = ts(data=misdatos,
                     start = 1956, frequency = 4)
class(miseriedetiempo)
```

```
## [1] "ts"
```

Creación del gráfico de la serie

Una vez transformado un objeto con datos en un objeto tipo serie de tiempo, estamos en condiciones de graficarlo.

```
plot(miseriedetiempo)
```



Modifiquemos el periodo de tiempo de inicio de la serie

Adicionalmente, podemos consultar que clase es la serie de tiempo que hemos creado (miseriedetiempo), que periodo de tiempo tiene y por supuesto, redefinir el inicio de análisis de la serie suponiendo que tenemos que corregir. Veamos:

```
class(miseriedetiempo) # clase times series
```

```
## [1] "ts"
```

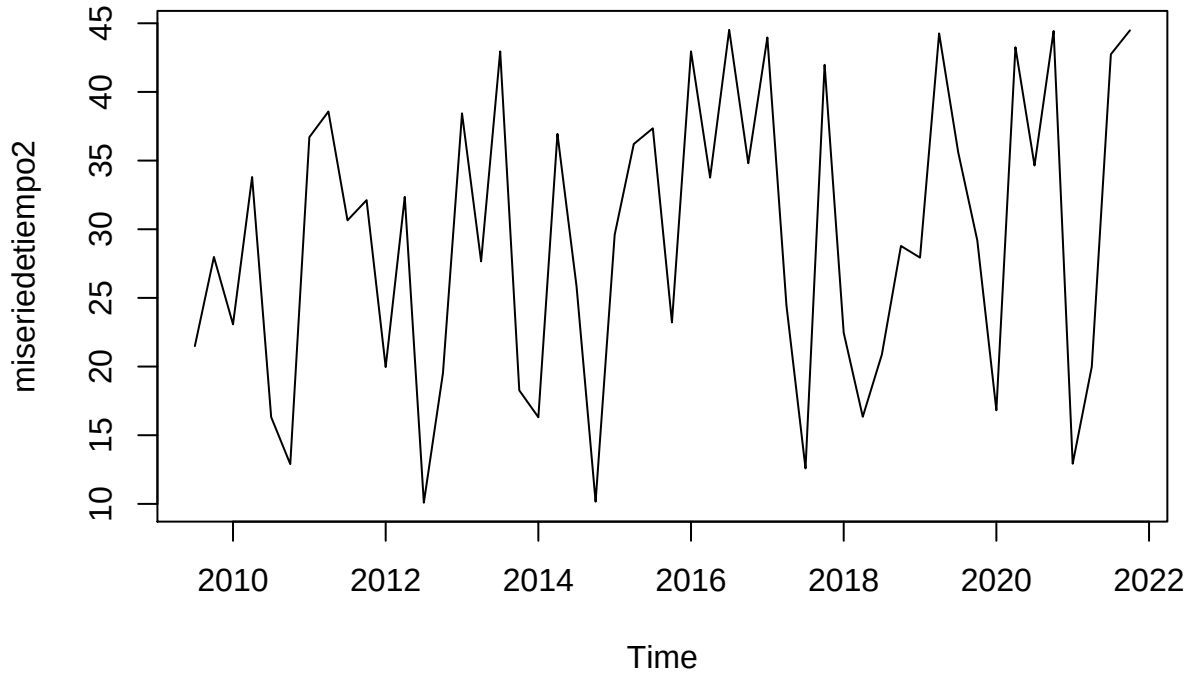
```
time(miseriedetiempo) # tiempo
```

```
##         Qtr1    Qtr2    Qtr3    Qtr4
## 1956 1956.00 1956.25 1956.50 1956.75
## 1957 1957.00 1957.25 1957.50 1957.75
## 1958 1958.00 1958.25 1958.50 1958.75
## 1959 1959.00 1959.25 1959.50 1959.75
## 1960 1960.00 1960.25 1960.50 1960.75
## 1961 1961.00 1961.25 1961.50 1961.75
## 1962 1962.00 1962.25 1962.50 1962.75
## 1963 1963.00 1963.25 1963.50 1963.75
## 1964 1964.00 1964.25 1964.50 1964.75
## 1965 1965.00 1965.25 1965.50 1965.75
## 1966 1966.00 1966.25 1966.50 1966.75
```

```
## 1967 1967.00 1967.25 1967.50 1967.75
## 1968 1968.00 1968.25
```

```
miseriedetiempo2 = ts(data=misdatos,
                      start=c(2009,3), frequency = 4) # ajusta el perido
```

```
plot(miseriedetiempo2) # graficando nuevamente el nuevo objeto creado
```

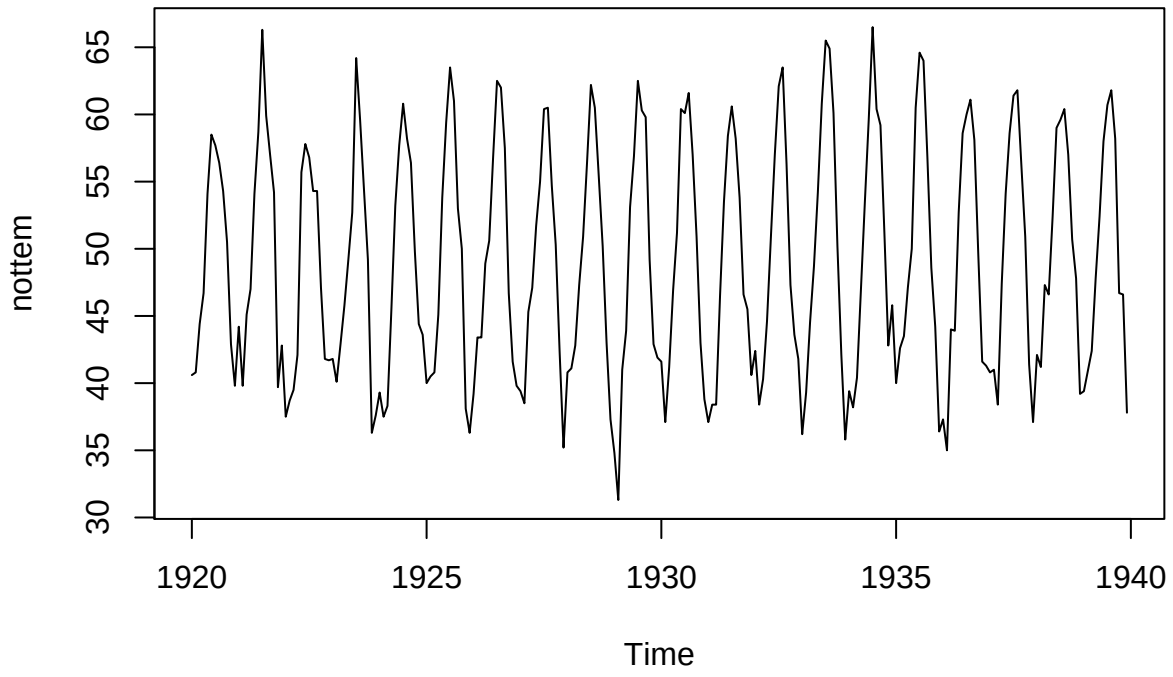


Trabajemos un poco más utilizando datos disponibles en RStudio, luego irémos a series más economicas.

Conjunto de datos Nottem

Nottem es una serie de tiempo que contiene el promedio de temperaturas en el castillo de Nottingham en grados Fahrenheit durante 20 años. Graficando la serie temporal tenemos:

```
plot(nottem) # graficando nottem
```



¿Qué podemos observar?

Que efectivamente la serie de tiempo que incluye observaciones asociadas al tiempo son altamente estacionales, ya que en Inglaterra existen estaciones y su variación evidencia ciclos. ¿Qué más puede decir esta serie? Comparte tus observaciones.

Utilicemos otros paquetes utiles en series de tiempo

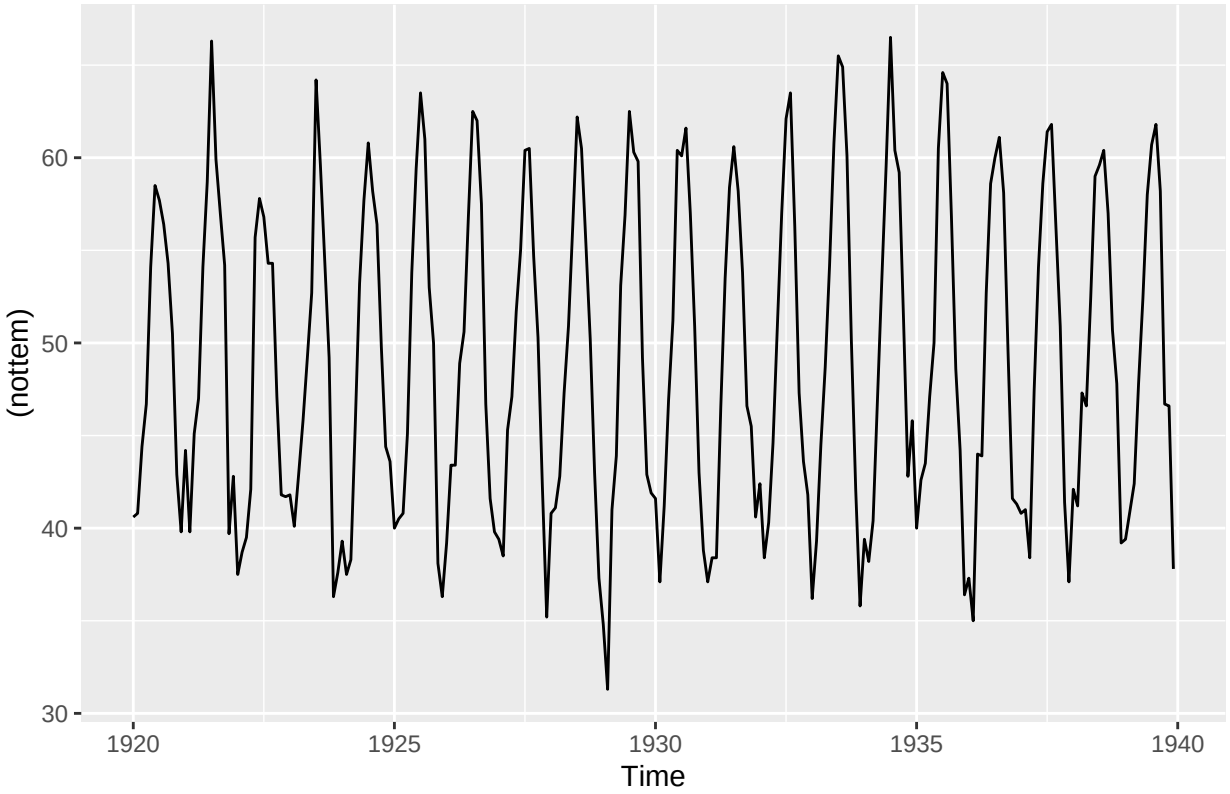
Existen dos paquetes muy útiles en el análisis de series de tiempo, estos paquetes son: 1) forecast que ya habíamos introducido antes y 2) ggplot2. Este ultimo paquete es un "sistema organizado de visualización de datos. Forma parte del conjunto de librerías llamado tidyverse. . . "[3]. Así una vez más lo sugerido es verificar si existe el paquete, de no existir instalarlo y si existe entonces cargar la librería respectiva. Y de no tener cargado ya el paquete forecast, hacerlo.

```
library(ggplot2)
library(forecast)
```

Si utilizamos el comando autoplot, nos grafica nuevamente la serie pero con la estructura del ggplot. Veamos:

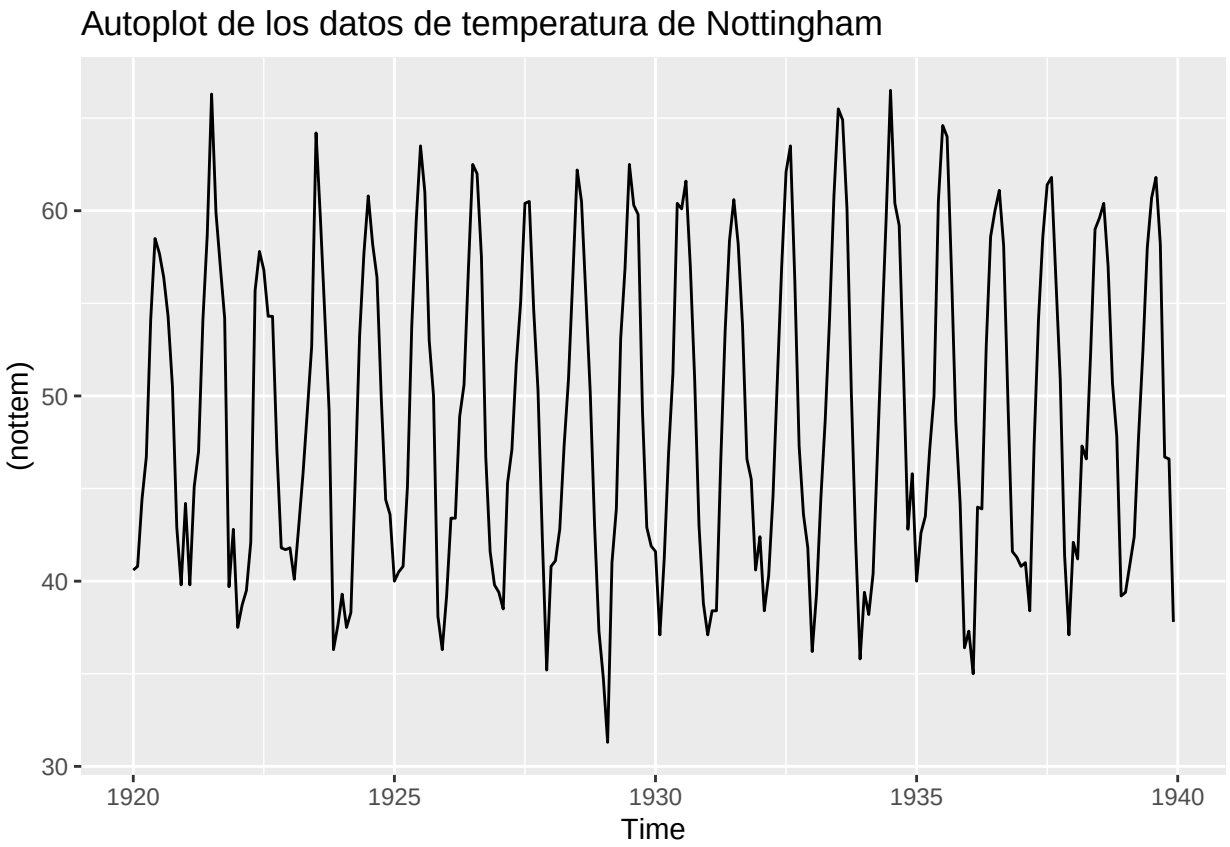
```
autoplot((nottem))
```

[3]Tomado de:https://bit.ly/2Uf55KX



Si le queremos agregar un título al gráfico agegamos ggtitle:

```
autoplot((nottem)) + ggtitle("Autoplot de los datos de temperatura de Nottingham")
```



Muy bien, hasta ahora hemos trabajado varias cosas que corresponden al pre-procesado de las Series de Tiempo: 1) Introducción a algunas librerías necesarias en el análisis de series de tiempo, 2) diferentes baterías para darle forma al tiempo en bases de datos, 3) introducimos nuestro primer objeto temporal a través de: a) creación de datos simulados, b) creación de una serie temporal. 4) creamos nuestro primer gráfico de serie, 5) modificamos el periodo de tiempo de inicio en una serie y finalmente 6)trabajamos con la serie Nottem en RStudio.

Ahora vamos a trabajar con datos de información omitida y atípicos.

Valores faltantes y atípicos (Outliers)

Este proceso corresponde a la identificación y tratamiento de valores faltantes y atípicos en las series temporales. Para esto vamos trabajar con otra base de datos diferente a la que hemos utilizado hasta ahora, por lo que vamos a importar los datos Rmissing en formato ts.

Previamente seleccionamos la ubicación la cual vamos a trabajar:

```
path <- ("C:/Users/Jose Oscar Henao M/Documents/")
```

Ahora estamos en condiciones de importar los datos

```
mydata <- read.csv("C:/Users/Jose Oscar Henao M/Documents/Rmissing.csv")
str(mydata)
```

```
## 'data.frame':    250 obs. of  2 variables:
##  $ X     : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ mydata: num  32.8 42.5 NA 32.2 55.6 ...
```

Una vez cargados los datos, queremos que se reconozca que la variable "mydata" es una serie de tiempo. Nota: No estamos especificando la frecuencia de esta serie de tiempo ;).

```
myts=ts(mydata$mydata)
myts
```

```
## Time Series:
## Start = 1
## End = 250
## Frequency = 1
##   [1]  32.801464  42.465485         NA  32.204058  55.557647  33.050864
##   [7]  43.401620  37.768318  22.844180  36.428877  28.496485  59.037881
##  [13]  36.544163  26.668135  41.325626  28.913199  38.595417  31.341447
##  [19]  34.547023         NA  30.499324  49.391323  43.976004  22.162741
##  [25]  19.439525  41.892407  30.321857  32.899878  17.686235  10.332791
##  [31]  31.612958  40.011275  35.378517  46.167222  26.903207  36.304821
##  [37]  23.408770  42.785841  31.919674  37.571226  33.907485  17.698917
##  [43]  19.931775  23.971169 999.000000  32.853670  33.012320  47.893249
##  [49]  33.961104  40.826518  34.389579  27.210322  41.815827         NA
##  [55]  49.711080  37.246486  34.472507  27.554913  37.976930  24.503481
##  [61]  33.941547  28.582326  17.945402  40.335543  32.103075  15.609346
##  [67]  38.637130  58.877558  42.178769  34.075469  29.208206  20.409934
##  [73]  23.682860  49.014566  59.160903  24.994359  37.321672  11.830421
##  [79]  49.907975  33.288427  25.900307  34.661099  38.170951  30.246685
##  [85]  45.001326  36.082827  38.969588  24.260726   8.619401  33.933167
##  [91]  30.158056  32.211135  46.688584  36.399098  27.266510  39.706101
##  [97]  48.560701 999.000000  31.011612  33.565184  41.850476  45.780926
## [103]  21.679404  32.340497  55.904896  17.349895  32.994516  36.155426
## [109]  47.089342  33.955275  36.563838  18.773382  28.077605  40.483324
## [115]  41.341771  32.907839  59.604911  20.989279  46.886734  53.931163
## [121]  44.662468  43.125045  25.800244  22.833920  51.397357  34.775922
## [127]  50.922532  36.430258  32.975690  37.659017  48.006323  49.901919
## [133]  20.619643  24.895206   4.682543  17.049461  45.618543  28.288209
## [139]  50.446258  34.983971  38.847283  32.301493  46.044574  21.739473
## [145]  16.457915  36.157602  35.773314  23.368300  34.220736  39.443674
## [151]  26.074044  28.599269  45.410516         NA  30.585004  23.405284
## [157]  36.949438  17.647508  22.991044  40.388899  30.654440  49.261182
## [163]  31.215505  30.462442  41.294423  28.046393  24.925970  23.934094
## [169]  41.690112  46.226476  40.741721 999.000000  26.455048  12.766929
## [175]  38.133315  61.653241  11.474054  41.835335  34.572898  59.921615
## [181]  53.072688  51.889866  43.700888  33.639492  24.988901  27.019376
## [187]  12.774882  21.001551  18.133113  48.563807  64.633075  29.362668
## [193]  45.478245  34.152629  50.573869  43.564419  57.644084  20.785408
## [199]  39.811707  51.854335  33.351763  23.242107  34.351879  28.113792
## [205]  33.952911  35.372668  38.059841  40.818440  45.768335  39.272128
## [211] 999.000000  36.665537  50.282893  34.561040  46.040830  39.936308
## [217]  39.873144  51.126396   2.683472  51.667975  41.336229  43.090450
## [223]  24.686842  52.300908  37.379943  19.043254  43.512121  54.236360
## [229]  33.557160  33.851597         NA  36.149460  32.985037  31.422766
## [235]  36.574851  29.648483  18.290954  38.154075  34.446452  12.037743
## [241]  23.581530  36.968395  50.747174  37.981389  28.693203  32.396009
## [247]  41.325484  30.017571  14.818111  45.403854
```

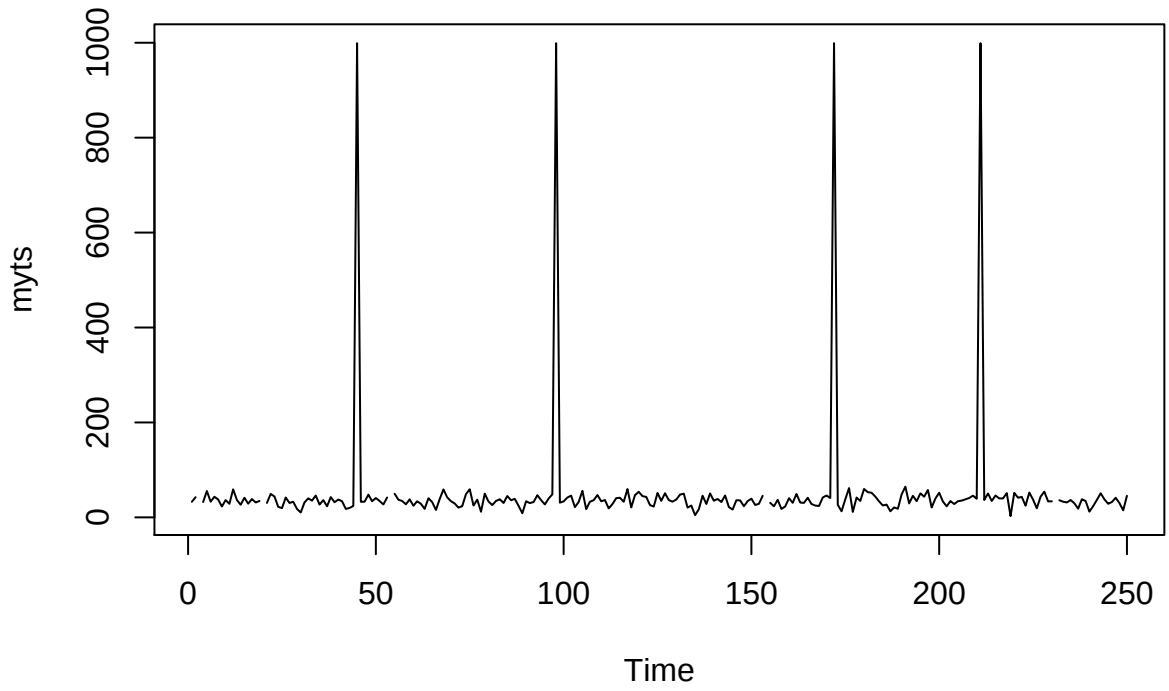
Como podemos observar en la tabla myts que hemos creado existen datos faltantes (NA) ¿Tenemos informa-

ción omitida NAs y Outliers? Para verificarlo utilizaremos dos herramientas fundamentales: 1) El resumen estadístico a través de "summary" y la gráfica.

```
summary(myts)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   2.683  28.078  34.573  50.710  42.465 999.000       5
```

```
plot(myts)
```



Como se pudo observar en el resumen estadístico tenemos 5 datos faltantes (NA) y posiblemente tenemos presencia de outliers por el rango de la serie de tiempo. Gráficamente podemos ver que tenemos lineas discontinuas (por los valores faltantes) y tenemos 4 puntos que representan datos atípicos.

Tratamiento de datos faltantes

Para el tratamiento de datos NA, vamos a utilizar la librería "zoo" y rellenarlos. Como anteriormente especificamos antes de cargar la librería debemos verificar que el paquete "zoo" ya se encuentre instalado.

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

la librería "zoo" cuenta con diferentes funciones: 1) locf: last observation carried forward (copia la última observación antes del NA) O 2) fill (rellenar con el valor que le pongamos). 3) otras opciones? les sugiero ver en "?zoo".

```
myts.NAlocf=na.locf(myts)
```

```
myts.NAfill=na.fill(myts,33)
```

Derección automática de outliers

Este proceso de detección lo vamos a trabajar con la librería "forecast" a través de la función "tsoutliers"

```
library(forecast)
```

```
myts1=tsoutliers(myts)
myts1
```

```
## $index
## [1]  45  98 172 211
##
## $replacements
## [1] 28.41242 39.78616 33.59838 37.96883
```

Esta función nos permite identificar los outliers que existen en la serie de tiempo y nos realiza una sugerencia de qué valores incorporar. Me estima cuales deberían ser esos valores.

Otra manera para detectar NAs desde la librería "forecas" es a través de la función "na.interp". Lo que hace esta función es rellenar NA con interpolación de los demás datos.Entiendase la interpolación como un proceso de usar valores conocidos para estimar valores desconocidos.

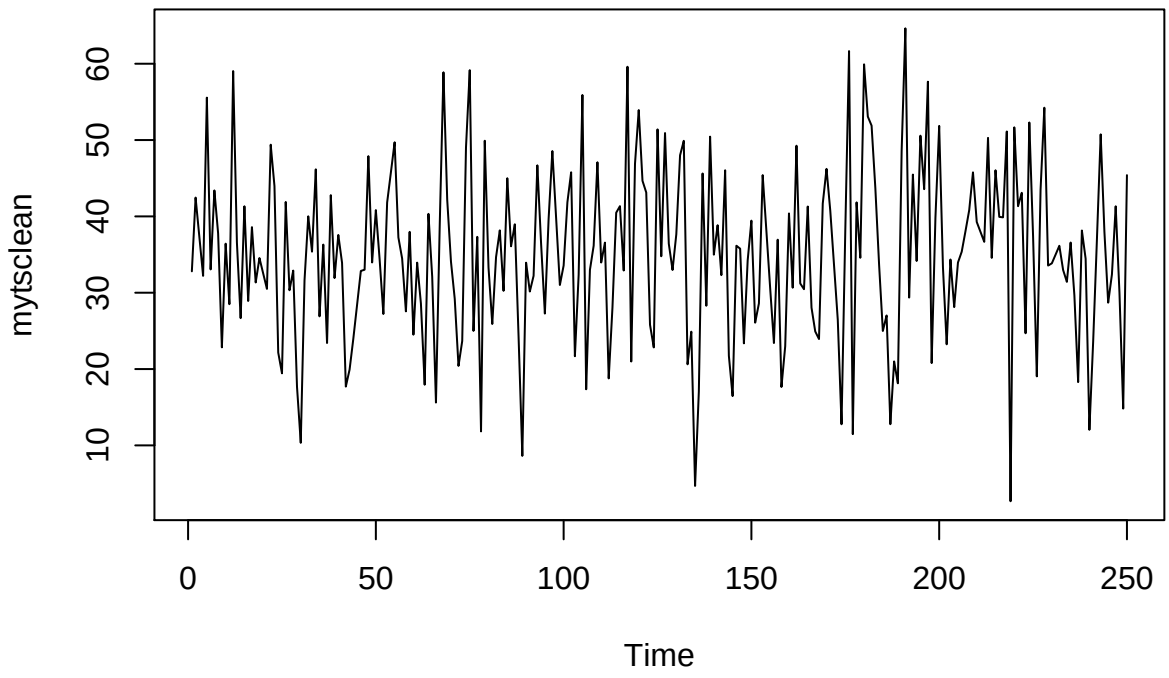
```
myts.NAinterp = na.interp(myts)
myts.NAinterp
```

```
## Time Series:
## Start = 1
## End = 250
## Frequency = 1
##   [1]  32.801464  42.465485  37.334771  32.204058  55.557647  33.050864
##   [7]  43.401620  37.768318  22.844180  36.428877  28.496485  59.037881
##  [13]  36.544163  26.668135  41.325626  28.913199  38.595417  31.341447
##  [19]  34.547023  32.523174  30.499324  49.391323  43.976004  22.162741
##  [25]  19.439525  41.892407  30.321857  32.899878  17.686235  10.332791
##  [31]  31.612958  40.011275  35.378517  46.167222  26.903207  36.304821
##  [37]  23.408770  42.785841  31.919674  37.571226  33.907485  17.698917
##  [43]  19.931775  23.971169 999.000000  32.853670  33.012320  47.893249
##  [49]  33.961104  40.826518  34.389579  27.210322  41.815827  45.763453
##  [55]  49.711080  37.246486  34.472507  27.554913  37.976930  24.503481
```

```
##  [61]  33.941547  28.582326  17.945402  40.335543  32.103075  15.609346
##  [67]  38.637130  58.877558  42.178769  34.075469  29.208206  20.409934
##  [73]  23.682860  49.014566  59.160903  24.994359  37.321672  11.830421
##  [79]  49.907975  33.288427  25.900307  34.661099  38.170951  30.246685
##  [85]  45.001326  36.082827  38.969588  24.260726   8.619401  33.933167
##  [91]  30.158056  32.211135  46.688584  36.399098  27.266510  39.706101
##  [97]  48.560701 999.000000  31.011612  33.565184  41.850476  45.780926
## [103]  21.679404  32.340497  55.904896  17.349895  32.994516  36.155426
## [109]  47.089342  33.955275  36.563838  18.773382  28.077605  40.483324
## [115]  41.341771  32.907839  59.604911  20.989279  46.886734  53.931163
## [121]  44.662468  43.125045  25.800244  22.833920  51.397357  34.775922
## [127]  50.922532  36.430258  32.975690  37.659017  48.006323  49.901919
## [133]  20.619643  24.895206   4.682543  17.049461  45.618543  28.288209
## [139]  50.446258  34.983971  38.847283  32.301493  46.044574  21.739473
## [145]  16.457915  36.157602  35.773314  23.368300  34.220736  39.443674
## [151]  26.074044  28.599269  45.410516  37.997760  30.585004  23.405284
## [157]  36.949438  17.647508  22.991044  40.388899  30.654440  49.261182
## [163]  31.215505  30.462442  41.294423  28.046393  24.925970  23.934094
## [169]  41.690112  46.226476  40.741721 999.000000  26.455048  12.766929
## [175]  38.133315  61.653241  11.474054  41.835335  34.572898  59.921615
## [181]  53.072688  51.889866  43.700888  33.639492  24.988901  27.019376
## [187]  12.774882  21.001551  18.133113  48.563807  64.633075  29.362668
## [193]  45.478245  34.152629  50.573869  43.564419  57.644084  20.785408
## [199]  39.811707  51.854335  33.351763  23.242107  34.351879  28.113792
## [205]  33.952911  35.372668  38.059841  40.818440  45.768335  39.272128
## [211] 999.000000  36.665537  50.282893  34.561040  46.040830  39.936308
## [217]  39.873144  51.126396   2.683472  51.667975  41.336229  43.090450
## [223]  24.686842  52.300908  37.379943  19.043254  43.512121  54.236360
## [229]  33.557160  33.851597  35.000528  36.149460  32.985037  31.422766
## [235]  36.574851  29.648483  18.290954  38.154075  34.446452  12.037743
## [241]  23.581530  36.968395  50.747174  37.981389  28.693203  32.396009
## [247]  41.325484  30.017571  14.818111  45.403854
```

Finalmente, paradesarrollar un proceso de limpieza de la serie de tiempo (NAs y Outliers) consiste en el uso de la función “tsclean” del paquete “forecast”.

```
mytsclean = tsclean(myts)
plot(mytsclean)
```




```
summary(mytsclean)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.683  28.157  34.567  35.025  41.830  64.633
```

Así estamos en condiciones de utilizar estas diferentes formas para limpiar nuestras bases de datos, en caso tal de requerirlo.

Veamos ahora una práctica para Argentina!!!

Analisis de la serie de tiempo del MERVAL (Argentina) a través de yahoo finance

El MERVAL, "índice emblemático de Argentina, busca medir el desempeño de las acciones de mayor tamaño y liquidez operadas en Bolsas y Mercados Argentinos (BYMA), que estén clasificadas como acciones locales. Los componentes del índice deben cumplir con los requisitos mínimos de tamaño y liquidez"[4]. Es un índice que mide el valor en pesos.

Las librerías que utilizaremos adicionalmente a las ya cargadas son: 1) quantmod, 2) xts y 3) reshape. - El paquete quanmod permite obtener, transformar y dibujar datos financieros de diversas fuentes. - El paquete xts (eXtensible Time Series) ayuda a leer y organizar datos como series temporales. - Paquete reshape permite la transformación de datos entre los formatos ancho y largo. Es una forma de redimensionar una base de datos (reestructurar y agregar datos de un data frame o data).

[4]Tomado de: https://bit.ly/3xYNYL6

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x chron::days()            masks lubridate::days()
## x dplyr::filter()          masks stats::filter()
## x chron::hours()           masks lubridate::hours()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x chron::minutes()         masks lubridate::minutes()
## x chron::seconds()         masks lubridate::seconds()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
## x chron::years()           masks lubridate::years()
```

```
library(lubridate)
library(quantmod)
```

```
## Loading required package: xts

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR
```

```
library(tseries)
library(xts)
library(zoo)
library(reshape)
```

```
##
## Attaching package: 'reshape'

## The following object is masked from 'package:dplyr':
##
##     rename
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths

## The following object is masked from 'package:lubridate':
##
##     stamp
```

Para nuestro análisis del Merval (Argentina) debemos cargar los datos. Anteriormente hemos visto que podemos cargar los datos en diferentes formatos o simularlos a través de datos uniformemente distribuidos. En esta ocasión cargaremos los datos desde "Yahoo finance".Les invito a consultar los datos en el siguiente link: https://es-us.finanzas.yahoo.com/quote/%5EMERV?p=%5EMERV

Para cargarlos utilizaremos el siguiente comando:

```
getSymbols("^MERV", src="yahoo", periodicity="daily", format="xts")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## Warning: ^MERV contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.

## [1] "^MERV"
```

Una vez cargado creamos los datos MERV1 que contiene los valores ajustados del Merval y una variable en formato xts que corresponde al tiempo.

```
MERV1 <- MERV$MERV.Adjusted
class(MERV1)
```

```
## [1] "xts" "zoo"
```

Verificando la clase de MERV1 identificamos que es "xts" en el tiempo y "zoo" en los valores ajustados.

Si queremos crear una función que pase de xts a data.frame podemos utilizar el siguiente comando:

```
xts_a_dataframe <- function(data_xts){
  df_t <- data.frame(fecha=(index(data_xts)),
                     value=coredata(data_xts))
  colnames(df_t) <- c("fecha", "valor")
  df_t
}
```

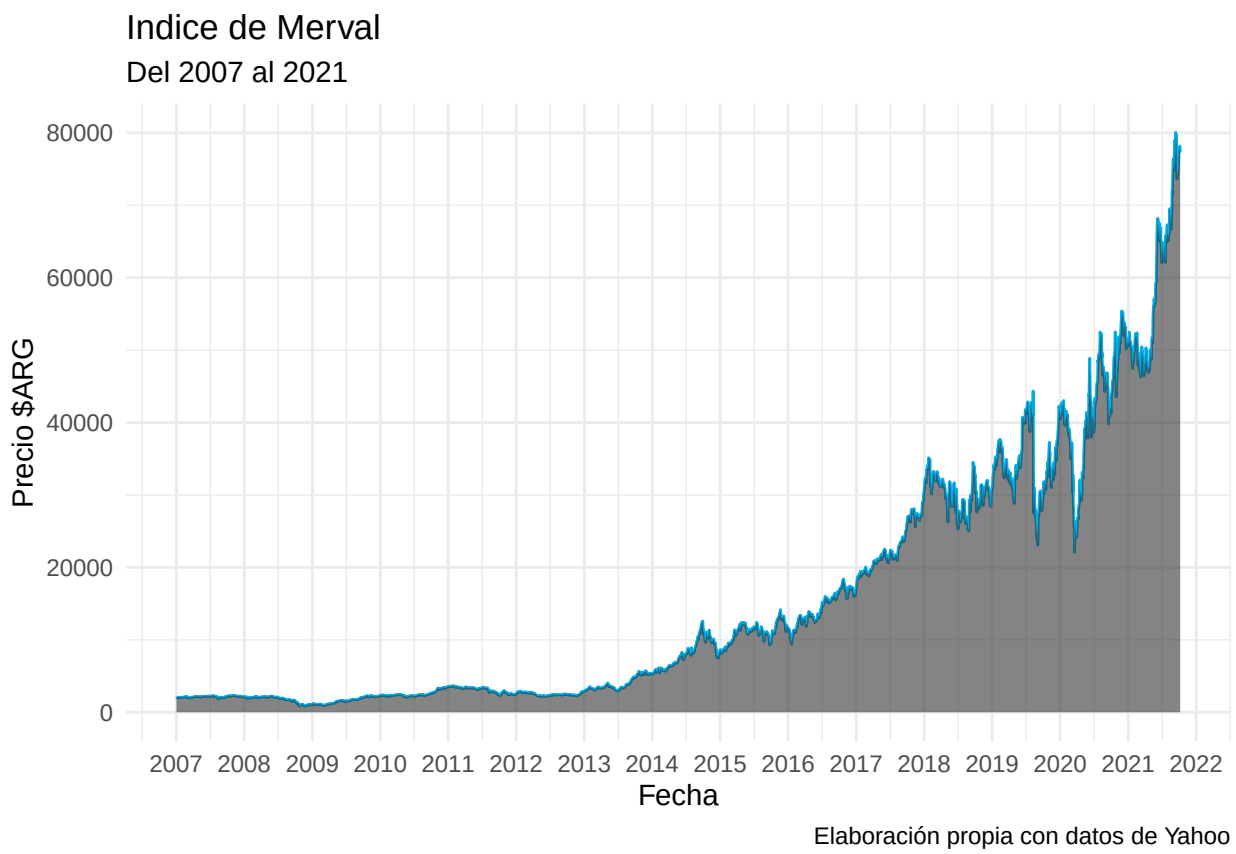
Esta función nos permitiría crear un data frame con dos variables “fecha” y “valor”.

A través de esta función estamos en condiciones de pasar los datos alojados en MERV1 que es de tipo zoo a un data frame. Aquí tambien corregimos los datos omitidos.

```
MERV1 <- na.omit(MERV1)
MERV1_df <- xts_a_dataframe(MERV1)
```

Bueno!!! visualicemos nuestros datos alojados en el data frame...

```
ggplot(MERV1_df, aes(y=valor, x=fecha))+
  geom_line(colour="#00AFEB")+
  geom_area(alpha=0.6)+
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")+
  theme_minimal()+
  labs(title="Indice de Merval",
       subtitle="Del 2007 al 2021",
       caption="Elaboración propia con datos de Yahoo",
       x="Fecha",
       y="Precio $ARG")
```



¿Qué podemos ver de esta gráfica? Que efectivamente es una serie que evidencia la presencia de una tendencia (creciente), ¿Que más se puede observar?

Ahora estamos en condiciones de ir más allá

Descomposición de la serie

a) Para hacerlo vamos a utilizar la función decompose, cabe aclarar que esta función solo puede aplicarse a un objeto en formato xts, no data.frame. Aquí lo que nos interesa en la descomposición es reconocer el movimiento de valores presentes en el MERVAL, así que lo primero que debemos hacer un ajuste a las fechas. Para garantizar que estamos trabajando con fechas, crearemos una serie Merval de clase "serie de tiempo".

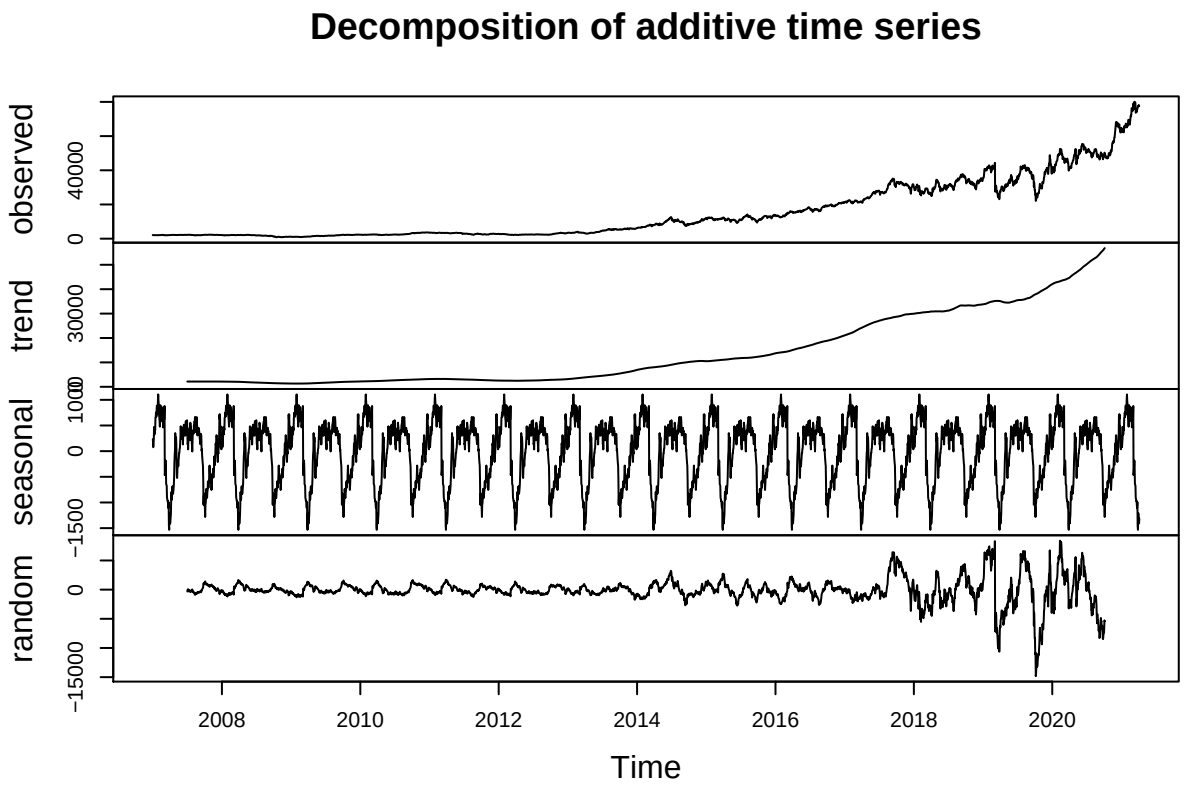
```
MERV1_ts <- ts(MERV1_df$valor, frequency = 252, start = c(2007,01))
class(MERV1_ts)
```

```
## [1] "ts"
```

Como se puede observar el objeto MERV1_ts es un objeto de tipo "ts".

Ahora veamos visualmente la descomposición de la serie MERVAL (Argentina): En 4 categorías 1) lo observado, 2) la tendencia, 3) la estacionalidad y 4 el componente aleatorio.

```
plot(decompose(MERV1_ts))
```



Ahora crearemos una variable nueva MERV tipo data frame que incluya lo observado, el componente estacional, la tendencia y el componente aleatorio.

```
MERV1_componente <- data.frame(fecha=time(MERV1_ts),
                               MERV1_ts=MERV1_ts,
```

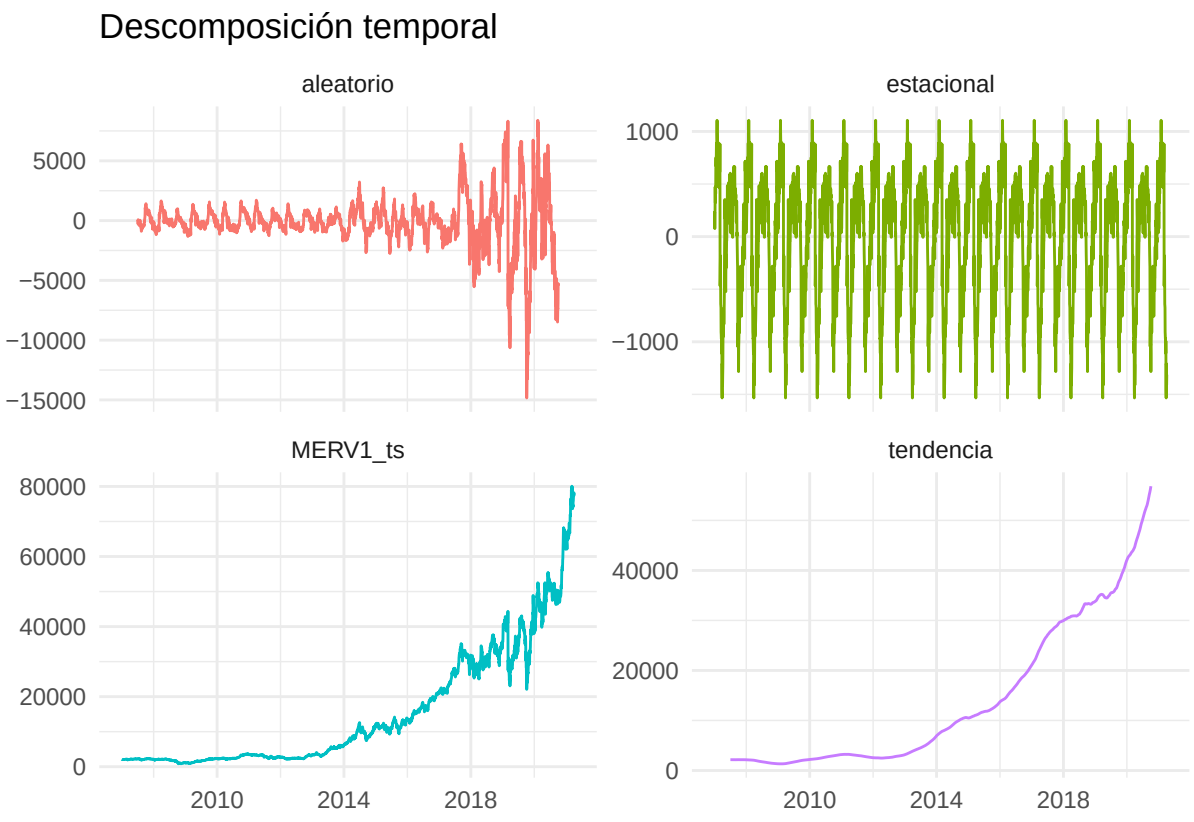
```
                          estacional=decompose(MERV1_ts)$seasonal,
                          tendencia=decompose(MERV1_ts)$trend,
                          aleatorio=decompose(MERV1_ts)$random)
```

Grafiquemos este data frame:

```
MERV1_componente %>%
  gather("id_var", "valores", -fecha) %>%
  ggplot(aes(x=fecha, y=valores))+
  geom_line(aes(color=id_var))+
  facet_wrap(.~id_var, scales = "free_y")+
  theme_minimal()+xlab("")+ylab("")+
  theme(legend.position = "none")+
  labs(title="Descomposición temporal")
```

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

## Warning: Removed 504 row(s) containing missing values (geom_path).
```



Veamos ahora algunas aplicaciones estadísticas en el análisis de series de tiempo.

Análisis estadístico de la serie

Agrupamiento de operador de retardos y diferencias en nuestra serie de tiempo.

```
retardos_MERV <- MERV1_df %>%
  mutate(fecha=ymd(fecha),
         retardo1=lag(valor, k=1),
         d1=diff(MERV1, differences = 1),
         d2=diff(MERV1, differences = 2),
         d1log=log(valor)-lag(log(valor)),
         tc_relativa=((valor/lag(valor,1))-1)*100)

class(retardos_MERV)
```

```
## [1] "data.frame"
```

Esta nos genera un data frame con diferentes variables: 1) retardo 1, 2) diferencia 1, 3) diferencia 2, 3) d1log = variación porcentual (dlog) y 4) cambio relativo en puntos porcentuales. Tenemos la serie con retados en un periodo, dif. 1, dif. 2, dif. log. y tc_relativa.

También podemos crear un data frame incorporando variables que especifique información vinculada al tiempo (dia, semana, trimestre, etc.)

```
MERV1_d<- MERV1_df %>%
  mutate(anio=format(fecha, "%Y"),
         dia_semana=format(fecha, "%A"),
         dia_semana_num=wday(fecha),
         dia_mes=format(fecha, "%d"),
         semana=format(fecha, "%W"),
         mes_tex = format(fecha, "%b"),
         mes_num=format(fecha, "%Q"),
         tc_MERV1=((valor/lag(valor, 1))-1)*100)

class(MERV1_d)
```

```
## [1] "data.frame"
```

Ahora analicemos estadísticamente los datos que hemos trabajado agrupados por temporalidad:

1. promedio con datos positivos y temporalidad (anual)

```
MERV1_d %>%
  group_by(anio) %>%
  summarise(promedio_posi=mean(tc_MERV1[tc_MERV1>0], na.rm = TRUE),
            promedio_nega=mean(tc_MERV1[tc_MERV1<0], na.rm = TRUE),
            promedio_total=mean(tc_MERV1, na.rm = TRUE))
```

```
## # A tibble: 15 x 4
##    anio  promedio_posi promedio_nega promedio_total
##    <chr>         <dbl>         <dbl>          <dbl>
##  1 2007          0.929        -1.10          0.0125
##  2 2008          1.62         -2.18         -0.236
##  3 2009          1.76         -1.58          0.339
##  4 2010          1.16         -1.15          0.185
##  5 2011          1.24         -1.38         -0.129
```

```
##  6 2012          1.27         -1.20          0.0745
##  7 2013          1.60         -1.31          0.282
##  8 2014          2.01         -2.03          0.231
##  9 2015          1.91         -1.71          0.156
## 10 2016          1.48         -1.32          0.169
## 11 2017          0.989        -0.939         0.243
## 12 2018          1.85         -1.78          0.0350
## 13 2019          2.23         -2.33          0.210
## 14 2020          2.29         -2.50          0.143
## 15 2021          1.58         -1.27          0.236
```

2. Evidenciar promedios según nuestro interés (positivo o negativo o total)

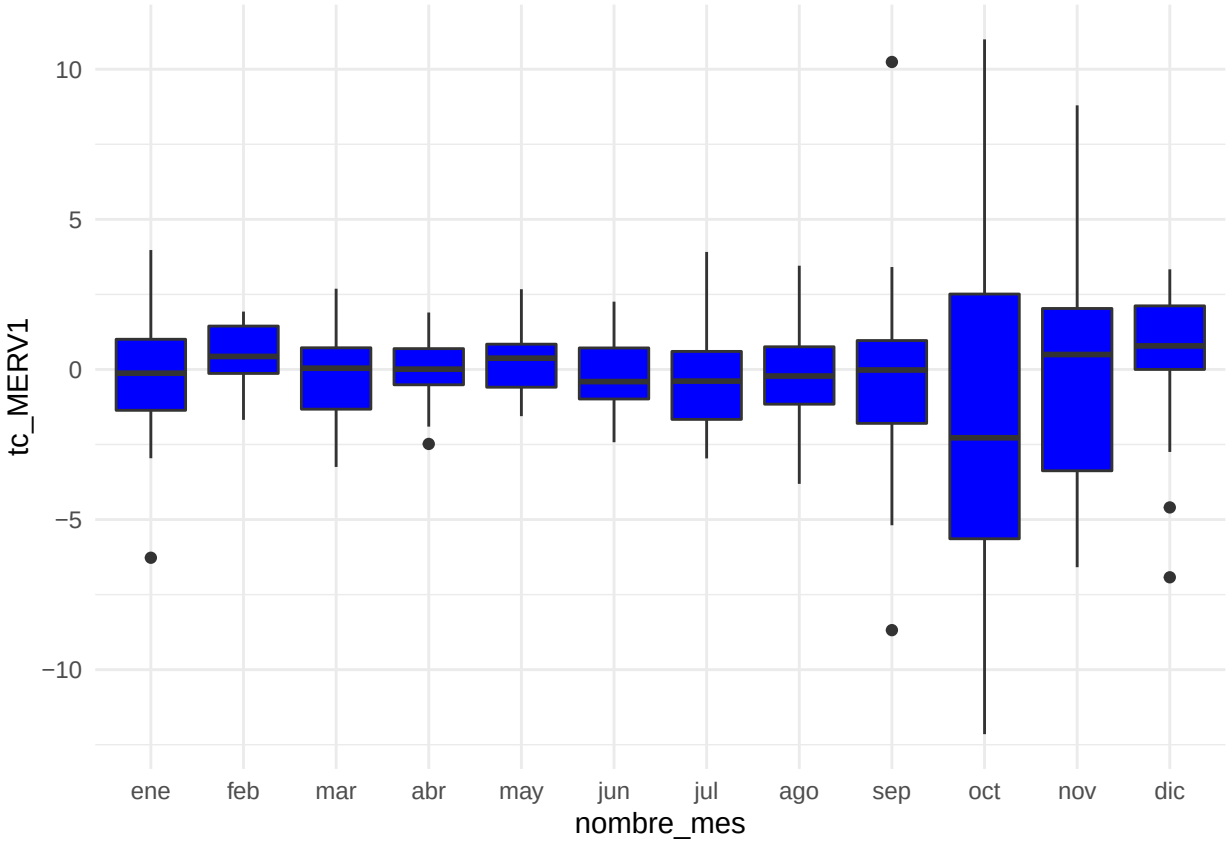
Por ejemplos ¿Qué dia de la semana es el mejor para el MERVAL?

```
MERV1_d %>%
  filter(anio == 2008) %>%
  group_by(dia_semana) %>%
  summarise(promedioDia=mean(tc_MERV1, na.rm = TRUE))
```

```
## # A tibble: 5 x 2
##   dia_semana promedioDia
##   <chr>            <dbl>
## 1 jueves          -0.114
## 2 lunes           -0.661
## 3 martes           0.104
## 4 miércoles       -0.393
## 5 viernes         -0.150
```

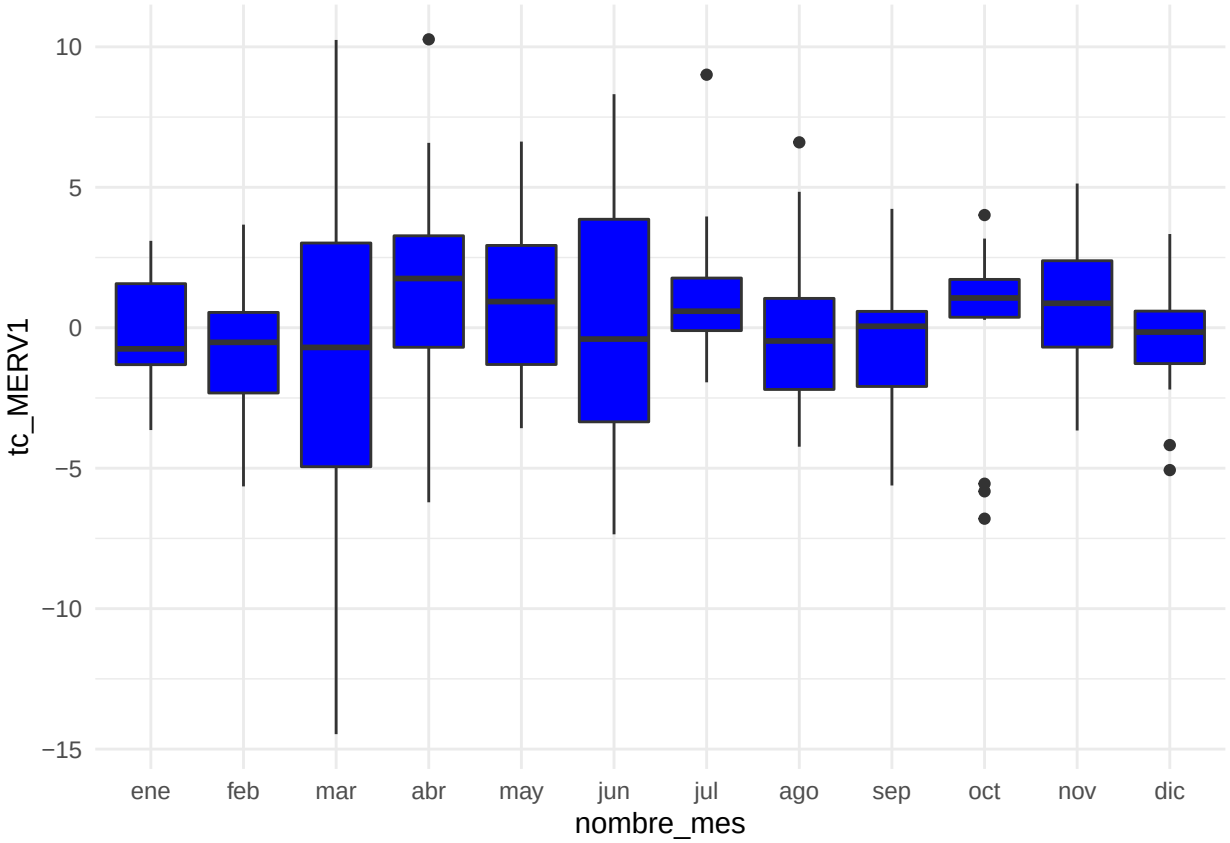
3. Tasa de variación en un periodo determinado Por ejemplo tasa de variación durante el 2008 a través de un grafica de caja. Aquí por ejemplo se observa el efecto de la crisis del 2008 en el MERVAL.

```
MERV1_d %>%
  mutate(nombre_mes = month(fecha, label = TRUE)) %>%
  filter (anio==2008) %>%
  ggplot(aes(x=nombre_mes, y=tc_MERV1))+
  geom_boxplot(fill="blue")+
  theme_minimal()
```



Veamos que sucedio en 2020 (Efecto COVID-19 en el MERVAL)

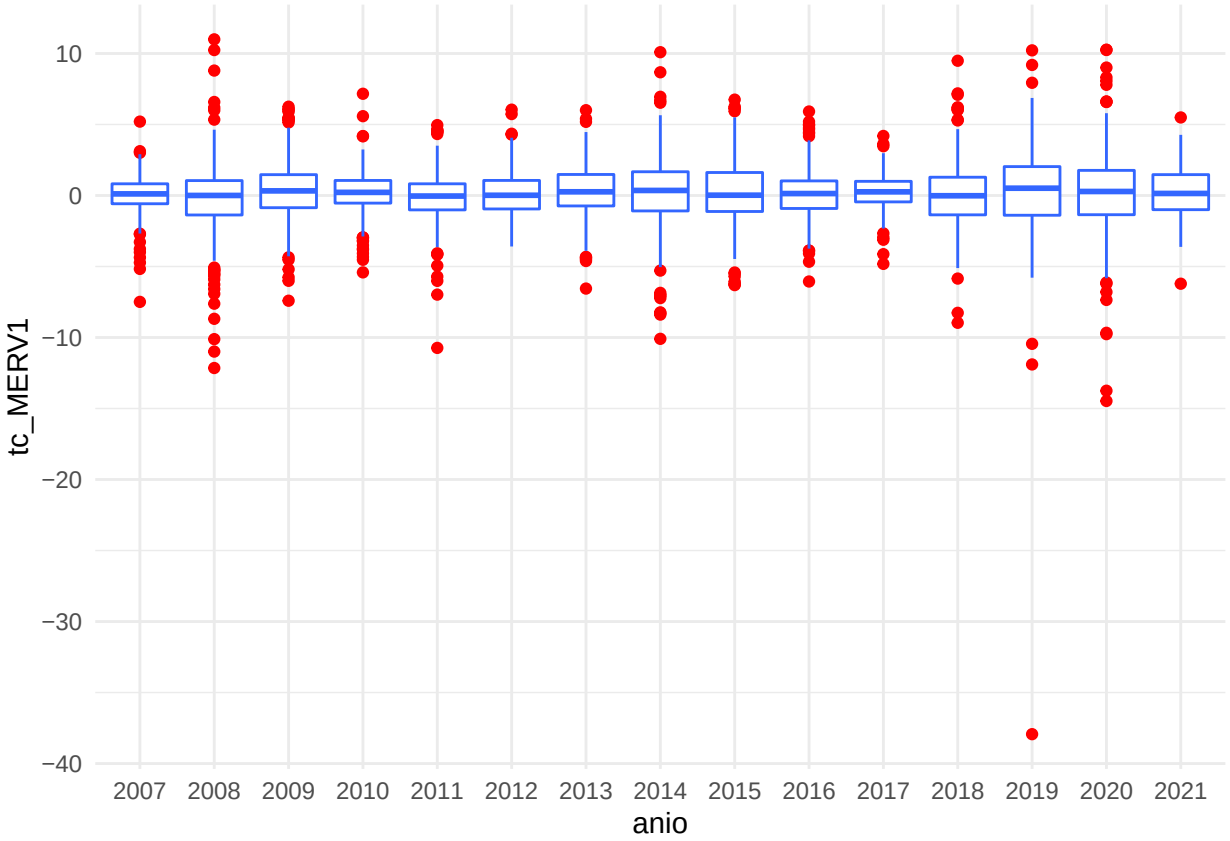
```
MERV1_d %>%
  mutate(nombre_mes = month(fecha, label = TRUE)) %>%
  filter (anio==2020) %>%
  ggplot(aes(x=nombre_mes, y=tc_MERV1))+
  geom_boxplot(fill="blue")+
  theme_minimal()
```



4. Distribución historica de las variables

```
MERV1_d %>%
  group_by(anio) %>%
  ggplot(aes(x=anio, y = tc_MERV1))+
  geom_boxplot(fill="white", colour="#3366FF",
               outlier.color = "red")+
  theme_minimal()
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```

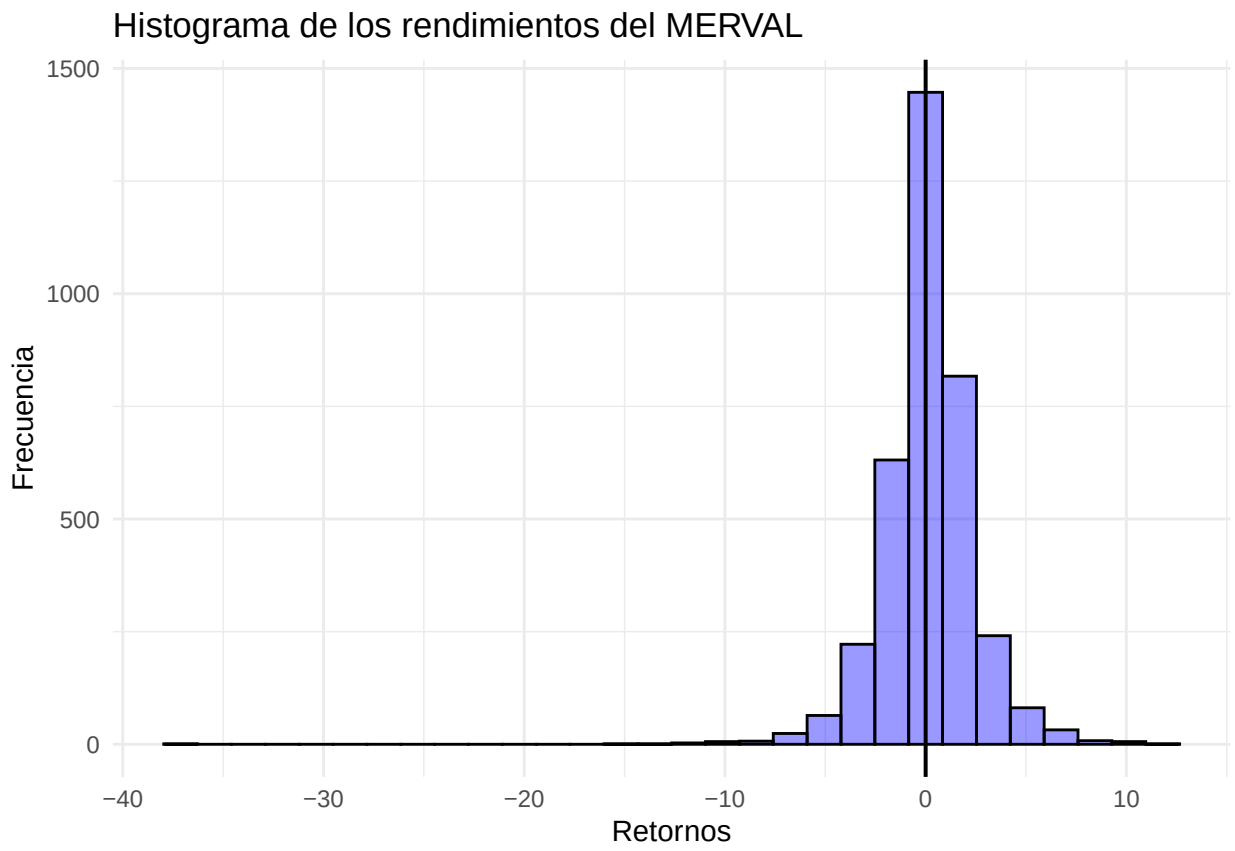


5. Histograma

```
ggplot(MERV1_d, aes(tc_MERV1))+
  geom_histogram(col="black", fill="blue", alpha=.4)+
  theme_minimal()+
  geom_vline(xintercept=0, color="black", size=0.7)+
  labs(title="Histograma de los rendimientos del MERVAL",
       x="Retornos", y="Frecuencia")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

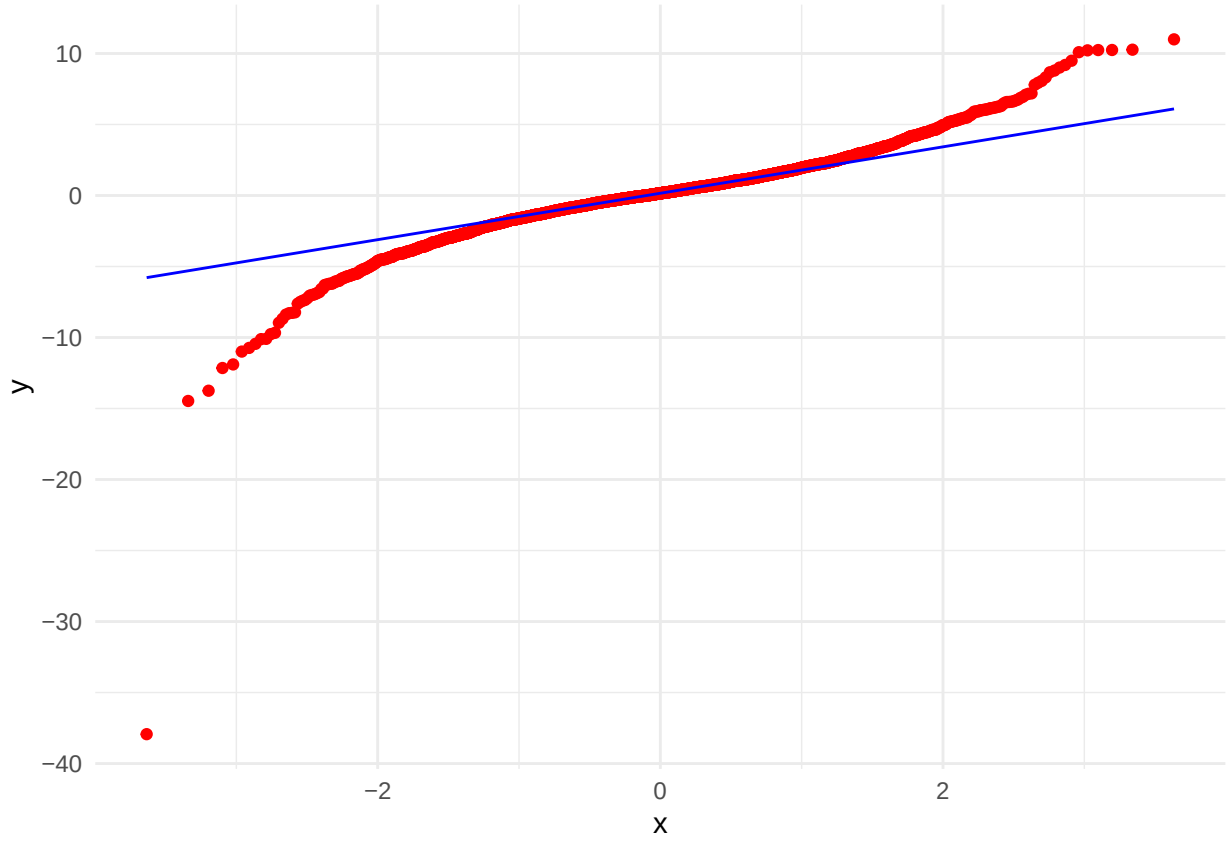


6. Gráfico QQ

```
ggplot(MERV1_d, aes(sample=tc_MERV1))+
  stat_qq(colour="red")+
  stat_qq_line(colour="blue")+
  theme_minimal()
```

```
## Warning: Removed 1 rows containing non-finite values (stat_qq).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_qq_line).
```



Datos atípicos del MERVAL

Primero debemos reconocerlos para esto utilizamos la función mutate y un resumen estadistico

```
MERV1_d %>%
  mutate(z_r=abs(scale(tc_MERV1)),
         mas_2_sd=factor((z_r>=3)*1, labels=c("No", "Atípico"))) %>%
  summarise(Obs=n(),
            porc=n()/nrow(MERV1_d))
```

```
##    Obs porc
## 1 3594    1
```

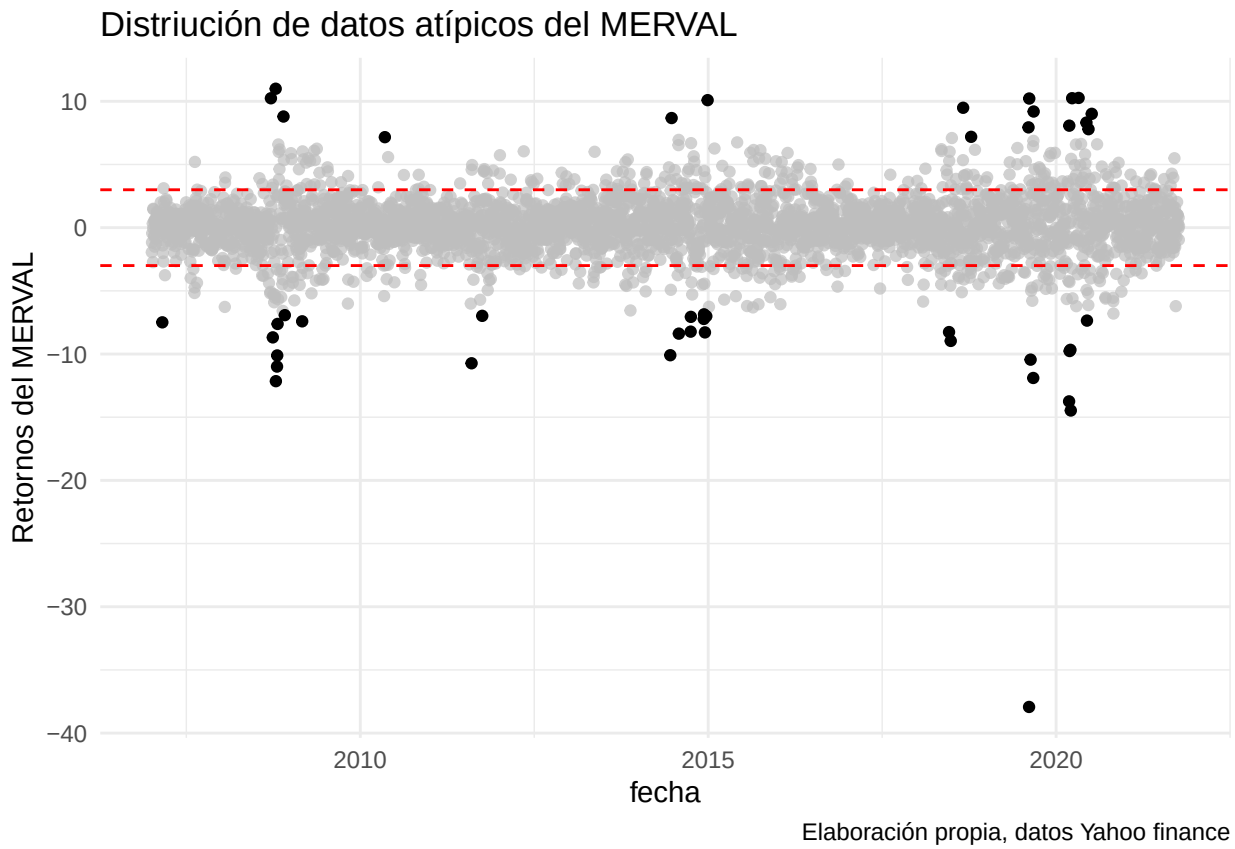
Gráficamente!!!

```
library(gghighlight)
```

```
MERV1_d %>%
  mutate(z_r=abs(scale(tc_MERV1)),
         mas_2_sd=factor((z_r>=3)*1, labels=c("No", "Atípico"))) %>%
  ggplot(aes(x=fecha, y=tc_MERV1))+
  geom_point()+
  gghighlight(mas_2_sd=="Atípico")+
```

```
  geom_hline(yintercept = c(3,-3), linetype="dashed", color="red")+
  theme_minimal()+
  labs(title="Distriución de datos atípicos del MERVAL",
       caption="Elaboración propia, datos Yahoo finance",
       y="Retornos del MERVAL",
       X="Fecha")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Bueno, con esto podemos dar por finalizado este primer documento introductorio para Series de Tiempo en RStudio para el curso de series de tiempo de la Pontificia Universidad Católica Argentina. Espero que este material que he diseñado les ayude en su aprendizaje, les invito a leer sobre los temas trabajados. Cualquier consulta les invito a escribirme al siguiente correo electrónico: josehenaomonje@uca.edu.ar