**Time series classification**

Time series classification uses supervised machine learning to analyze multiple labeled classes of time series data and then predict or classify the class that a new data set belongs to. This is important in many environments where the analysis of sensor data or financial data might need to be analyzed to support a business decision. Accuracy of classification is critical in these situations, so data scientists work hard to ensure that their time series classifiers are as accurate as possible.

as.ts and is.ts respectively coerce a vector into a time-series and test whether an object is a time series.

Time series have methods associated with the generic print and plot functions. The argument calendar to the print method can be used to enable/disable the display of information about month names, quarter names or year when printing.

Example is the amount of rainfall in a region at different months of the year. R language uses many functions to create, manipulate and plot the time series data. The data for the time series is stored in an R object called time-series object. It is also a R data object like a vector or data frame.

The time series object is created by using the ts() function.

Syntax

The basic syntax for ts() function in time series analysis is −

timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used −

• data is a vector or matrix containing the values used in the time series.

• start specifies the start time for the first observation in time series.

• end specifies the end time for the last observation in time series.

• frequency specifies the number of observations per unit time.

Except the parameter "data" all other parameters are optional

The function ts is used to create time-series objects. These are vector or matrices with class of "ts" (and additional attributes) which represent data which has been sampled at equispaced points in time. In the matrix case, each column of the matrix data is assumed to contain a single (univariate) time series.

The value of argument frequency is used when the series is sampled an interal number of times in each unit time interval. For example, one could use a value of 7 for frequency when the data are sampled daily, and the natural time period is a week, or 12 when the data are sampled monthly and the natural time period is a year.

start and end can either be integers which correspond to natural time units, or vectors of two integers, which give a natural time unit and a (1-based) number of samples into the time unit.

Consider the annual rainfall details at a place starting from January 2012. We create an R time series object for a period of 12 months and plot it.

Code to run in R

# Get the data points in form of a R vector.

rainfall <- c(799,1174.8,865.1,1334.6,635.4,918.5,685.5,998.6,784.2,985,882.8,1071)

# Convert it to a time series object.

rainfall.timeseries <- ts(rainfall,start = c(2012,1),frequency = 12)

# Print the timeseries data.

print(rainfall.timeseries)

# Give the chart file a name.

png(file = "rainfall.png")

# Plot a graph of the time series.

plot(rainfall.timeseries)

# Save the file.

dev.off()

After this again plot to get chart

plot(rainfall.timeseries)

Let’s take the example of the COVID-19 pandemic situation. Taking the total number of positive cases of COVID-19 cases weekly from 22 January 2020 to 15 April 2020 the world in data vector

# Weekly data of COVID-19 positive cases from

# 22 January, 2020 to 15 April, 2020

x <- c(580, 7813, 28266, 59287, 75700,

    87820, 95314, 126214, 218843, 471497,

    936851, 1508725, 2072113)

# library required for decimal\_date() function

library(lubridate)

# output to be created as png file

png(file ="timeSeries.png")

# creating time series object

# from date 22 January, 2020

mts <- ts(x, start = decimal\_date(ymd("2020-01-22")),

                            frequency = 365.25 / 7)

# plotting the graph

plot(mts, xlab ="Weekly Data",

        ylab ="Total Positive Cases",

        main ="COVID-19 Pandemic",

        col.main ="darkgreen")

# saving the file

dev.off()

Multivariate Time Series is creating multiple time series in a single chart. Taking data of total positive cases and total deaths from COVID-19 weekly from 22 January 2020 to 15 April 2020 in a data vector.

# Weekly data of COVID-19 positive cases and

# weekly deaths from 22 January, 2020 to

# 15 April, 2020

positiveCases <- c(580, 7813, 28266, 59287,

                75700, 87820, 95314, 126214,

                218843, 471497, 936851,

                1508725, 2072113)

deaths <- c(17, 270, 565, 1261, 2126, 2800,

            3285, 4628, 8951, 21283, 47210,

            88480, 138475)

# library required for decimal\_date() function

library(lubridate)

# output to be created as png file

png(file="multivariateTimeSeries.png")

# creating multivariate time series object

# from date 22 January, 2020

mts <- ts(cbind(positiveCases, deaths),

start = decimal\_date(ymd("2020-01-22")),

                    frequency = 365.25 / 7)

# plotting the graph

plot(mts, xlab ="Weekly Data",

    main ="COVID-19 Cases",

    col.main ="darkgreen")

# saving the file

dev.off()

**Forecasting** can be done on time series using some models present in R. In this example, Arima automated model is used. To know about more parameters of [arima()](https://www.geeksforgeeks.org/time-series-analysis-using-arima-model-in-r-programming/) function, use the below command.

help("arima")

In the below code, forecasting is done using the forecast library and so, installation of the forecast library is necessary.

# Weekly data of COVID-19 cases from

# 22 January, 2020 to 15 April, 2020

x <- c(580, 7813, 28266, 59287, 75700,

    87820, 95314, 126214, 218843,

    471497, 936851, 1508725, 2072113)

# library required for decimal\_date() function

library(lubridate)

# library required for forecasting

library(forecast)

# output to be created as png file

png(file ="forecastTimeSeries.png")

# creating time series object

# from date 22 January, 2020

mts <- ts(x, start = decimal\_date(ymd("2020-01-22")),

                            frequency = 365.25 / 7)

# forecasting model using arima model

fit <- auto.arima(mts)

# Next 5 forecasted values

forecast(fit, 5)

# plotting the graph with next

# 5 weekly forecasted values

plot(forecast(fit, 5), xlab ="Weekly Data",

ylab ="Total Positive Cases",

main ="COVID-19 Pandemic", col.main ="darkgreen")

# saving the file

dev.off()

4945137