Lab 1 - Time Series Data in R

Benjamin M. Taylor, Kevin Hayes

January 20, 2023

1 Pre-loaded Time Series Datasets in R

[ ]:
```
options(repr.plot.width=15, repr.plot.height=15) # makes plots bigger in the␣
↪webpage
```

The statistical software package R houses several (currently 30) preloaded time series data sets within the standard build. These are very useful for teaching purposes and are available as the R time series objects:

airmiles	AirPassengers	austres	BJsales	BJsales.lead
co2	discoveries	EuStockMarkets	fdeaths	freeny.y
JohnsonJohnson	LakeHuron	ldeaths	lh	lynx
mdeaths	nhtemp	Nile	nottem	presidents
Seatbelts	sunspot.month	sunspot.year	sunspots	treering
UKDriverDeaths	UKgas	USAccDeaths	uspop	WWWusage

To get information on any of these data sets simply type a question mark followed by the name of the corresponding R object at the R prompt. For example typing

```
?co2
```

will open a help file that: provides a brief description of the data set; clarifies the time series structure/format of the data; and identifies the original source of the time series data set. **Try it now**. As is the case with any R object, simply typing its name into the R prompt will reveal the contents and architecture of the R object in question:

[ ]:
```
co2
```

[ ]:
```
class(co2)
```

The 468 values in the R time series object `co2` are implicitly linked to 468 monthly recording times beginning in January 1959 and ending in December 1997, implying a sampling frequency of 12 values per year. By defining a time series class, R obviates the need to construct and manipulate the collection times as a second vector of values. The R graphics and programming environment will automatically recognise and apply procedures suitable for time series data when functions are actioned on `co2`. The most obvious illustration of this paradigm would be generic plotting function command. Try the following commands:

```
plot(co2, ylab = expression("Atmospheric concentration of CO"[2]), las = 1)
title(main = "co2 data set")
```

which produces Figure:

[ ]:
```
plot(co2, ylab = expression("Atmospheric concentration of CO"[2]), las = 1)
title(main = "co2 data set")
```

Over the coming weeks you are encouraged to be familiarise yourself with these data sets. Specifically, inspect the datasets and their help entries and plot the data. The following R commands are useful for extracting the structure of a time series object. Try the following:

```
frequency(co2)
cycle(co2)
time(co2)
```

[ ]:
```
frequency(co2)
```

[ ]:
```
cycle(co2)
```

[ ]:
```
time(co2)
```

The R command `window(x)` is is a generic function which extracts the subset of the object `x` observed between the times start and end. If a frequency is specified, the series is then re-sampled at the new frequency. Try the following:

```
window(co2, start = c(1959,7), end = c(1962,12))
window(co2, frequency = 4)
(co2.yr = window(co2, frequency = 1))          # a TS of yearly totals
```

[ ]:
```
window(co2, start = c(1959,7), end = c(1962,12))
```

[ ]:
```
window(co2, frequency = 4)
```

[ ]:
```
(co2.yr = window(co2, frequency = 1))          # a TS of yearly totals
```

We can enhance the time series plot of `co2` using some additional plotting function command. Try the following commands to add a LOWESS smother to the time series plot above:

```
co2.y = c(co2.yr)    # the nesting c( ) command returns a regular vector
co2.x = c(time(co2.yr))
lines(co2.x, co2.y, type = "o", lwd = 3, col = "red")

co2.lowess = lowess(co2, f=1/3)        # the classic LOWESS smoother
lines(co2.lowess$x, co2.lowess$y, lwd = 2, col = "blue")
```

[ ]:
```
co2.y = c(co2.yr)    # the nesting c( ) command returns a regular vector
co2.x = c(time(co2.yr))
plot(co2)
lines(co2.x, co2.y, type = "o", lwd = 3, col = "red")
```

[ ]:
```
co2.y = c(co2.yr)    # the nesting c( ) command returns a regular vector
co2.x = c(time(co2.yr))
```

```r
plot(co2)
lines(co2.x, co2.y, type = "o", lwd = 3, col = "red")

co2.lowess = lowess(co2, f=1/3)        # the classic LOWESS smoother
lines(co2.lowess$x, co2.lowess$y, lwd = 2, col = "blue")
```

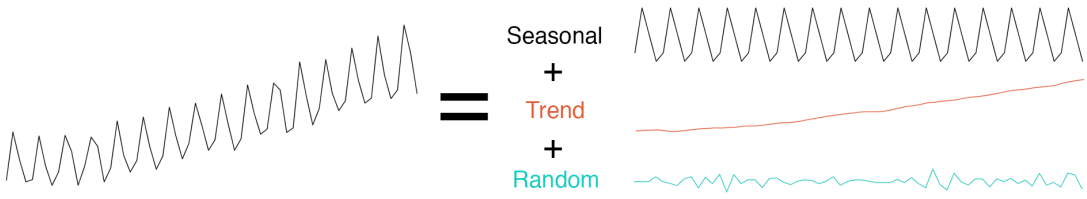
2 Common Features in Time Series

Some important questions to first consider when first looking at a time series are:

- Is there a trend, meaning that the measurements tend to increase (or decrease) on gradually over time?
- Is there seasonality, meaning that there is a regularly repeating pattern of highs and lows related to calendar time such as seasons, quarters, months, days of the week, and so on?
- Is there a long-run cycle or period unrelated to seasonality factors?
- Is there constant variance over time, or is the variance non-constant?
- Are there any abrupt changes to either the level of the series or the variance?
- Are there outliers? And if so, what type of outliers are they??
- Are we considering a single time series record, or multiple time series collected at coincident times?
- What is the objective of the statistical analysis? (i) Modelling and characterising the underlying temporal structure(s) of the data; (ii) Forecasting future values of the series along with appropriate forecasting errors; (iii) Both (i) and (ii).

3 Time-Series Decomposition

The classical additive framework for time series data is to consider separate seasonal, trend, and random components as follows:



Time series decomposition

The function `decompose()` decomposes a time series into seasonal, trend and irregular components using moving averages. (Deals with additive or multiplicative seasonal component.) Alternatively, the function `stl()` decomposes a time series into seasonal, trend and irregular components using lowess smoothing.

Try the following commands and compare the results.

```r
plot(decompose(co2))

co2.decomp = decompose(co2)    # saves and plots
```

```
names(co2.decomp)
plot(co2.decomp)

fit <- stl(log(co2), s.window = 20, t.window = 20)
plot(fit)
```

[ ]:
```
plot(decompose(co2))
```

[ ]:
```
co2.decomp = decompose(co2)    # saves and plots
names(co2.decomp)
```

[ ]:
```
plot(co2.decomp) # so the plot can be recalled at a later date
```

[ ]:
```
fit = stl(log(co2), s.window = 20, t.window = 20)
plot(fit)
```

The very versatile function `filter()` applies linear filtering to a univariate time series or to each series separately of a multivariate time series. Try:

```
plot(co2)
lines(filter(co2, filter = rep(1,6)/6), lwd = 2, col = "red")
lines(filter(co2, filter = c(1,rep(2,10),1)/22),  lwd = 2, col = "blue")
```

[ ]:
```
plot(co2)
lines(filter(co2, filter = rep(1,6)/6), lwd = 2, col = "red")
lines(filter(co2, filter = c(1,rep(2,10),1)/22),  lwd = 2, col = "blue")
```

The function `monthplot()` plots seasonal (or other) sub-series of a time series.

Try:

```
op = par(mfrow = c(2,2))
monthplot(co2, ylab = "data", cex.axis = 0.8)
monthplot(fit, choice = "seasonal", cex.axis = 0.8)
monthplot(fit, choice = "trend", cex.axis = 0.8)
monthplot(fit, choice = "remainder", type = "h", cex.axis = 0.8)
par(op)
```

[ ]:
```
op = par(mfrow = c(2,2))
monthplot(co2, ylab = "data", cex.axis = 0.8)
monthplot(fit, choice = "seasonal", cex.axis = 0.8)
monthplot(fit, choice = "trend", cex.axis = 0.8)
monthplot(fit, choice = "remainder", type = "h", cex.axis = 0.8)
par(op)
```

4 Average Monthly Temperatures at Nottingham, 1920–1939

The R time series object `nottem` contains monthly average air temperatures at Nottingham Castle in degrees Fahrenheit for 20 years, (source: Anderson, 1976). Plot the data. Just like the co2 data,

this series reports monthly averages. As with the co2 data, these data also display a cyclical / seasonal pattern. Unlike the co2 data, these data don't show any discernible trend over time time. Carry out a time-series decomposition of these data.

- Anderson (1976) Time Series Analysis and Forecasting: The Box-Jenkins approach. Butterworths. Series R.

[ ]:

5 Monthly Airline Passenger Numbers 1949-1960

The R time series object `AirPassengers` contains monthly monthly totals of international airline passengers, 1949 to 1960 (source: Box and Jenkins, 1976). Use the function `decompose()` and subsequently the function `stl()` to decomposes this time series into seasonal, trend and irregular components. Contrast the results.

- Box, G. E. P., Jenkins, G. M. and Reinsel, G. C. (1976) Time Series Analysis, Forecasting and Control. Third Edition. Holden-Day. Series G.

[ ]:

6 Multiple Time Series R Objects

6.1 Daily Closing Prices of Major European Stock Indices, 1991–1998

The R time-series object `EuStockMarket` contains the daily closing prices of major European stock indices: Germany DAX (Ibis), Switzerland SMI, France CAC, and UK FTSE. The data are sampled in business time, i.e., weekends and holidays are omitted. This is a multivariate time series with 1860 observations on 4 variables. The object is of class `mts`.

- The data were provided to the R environment by Erste Bank AG, Vienna, Austria.

Try each of the following R commands in turn, and at each step speculate what is happening.

```
plot(EuStockMarkets)
plot(log(EuStockMarkets))
plot(diff(log(EuStockMarkets)))
```

[ ]:

6.2 Road Casualties in Great Britain 1969–84

The R time-series object `UKDriverDeaths` is a time series giving the monthly totals of car drivers in Great Britain killed or seriously injured Jan 1969 to Dec 1984. Compulsory wearing of seat belts was introduced on 31 Jan 1983. The R time-series object `Seatbelts` is a multivariate time-series object with more information on the same problem.

- Harvey, A. C. and Durbin, J. (1986). The effects of seat belt legislation on British road casualties: A case study in structural time series modelling. Journal of the Royal Statistical Society series A, 149, 187–227.

Try each of the following R commands in turn, and at each step speculate what is happening.

```
plot(Seatbelts)
plot(UKDriverDeaths)
plot(stl(UKDriverDeaths, s.window=12))
```

[ ]:

7 Formative Assessment - Authenticating your R Plots

Write your own code to reproduce the Figure below as best you can. (Feel free to improve on the plot if you wish.) Include your name and ID number in the main title of the plot. You will be required to authenticate your work in this manner in future assessments.