A short introduction to R (with a glance at time series)

Stefano Tonellato

Department of Economics
Ca' Foscari University of Venice

THE CONTRACTOR IS ACTING UNDER A FRAMEWORK CONTRACT CONCLUDED WITH THE COMMISSION

devstat

Eurostat

- R is a software environment for data manipulation, calculation and graphical display.
- A lot of resources can be found at `http://cran.r-project.org/`
- R operates on *objects*, i.e. data structures that may be (numeric or character) vectors, arrays, data.frames, lists...

```
> 3+2
[1] 5
> sin(3+2)
[1] -0.9589243
> 7/sin(3+2)
[1] -7.299846
```


```
> x <- 3
> y <- 2
> z <- sin(x+y)
> z
[1] -0.9589243
> w <- 7/sin(3+2)
> w
[1] -7.299846
```

```
> x <- c(1,3,9,49049,23)
> x
[1]     1     3     9 49049    23
> y <- 1:10
> z <- seq(12, 20, by=2)
> z
[1] 12 14 16 18 20
> w <- rep(10,5)
> w
[1] 10 10 10 10 10
```


```
> z + w          # sum element by element
[1] 22 24 26 28 30
> z / w          # divide element by element
[1] 1.2 1.4 1.6 1.8 2.0
> z * w          # multiply element by element
[1] 120 140 160 180 200
```


```
> t(z) %*% w          # compute z'*w
     [,1]
[1,]  800
> z %*% t(w)          # compute z*w'
     [,1] [,2] [,3] [,4] [,5]
[1,]  120  120  120  120  120
[2,]  140  140  140  140  140
[3,]  160  160  160  160  160
[4,]  180  180  180  180  180
[5,]  200  200  200  200  200
```

```
> X <- matrix(1:12, nrow=4)
> X
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> Y <- matrix(1:12, nrow=4, byrow=T)
> Y
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

```
> Z <- X * Y          # element by element matrix product
> Z
     [,1] [,2] [,3]
[1,]    1   10   27
[2,]    8   30   60
[3,]   21   56   99
[4,]   40   88  144
> W <- t(X) %*% Y          # matrix product
> W
     [,1] [,2] [,3]
[1,]   70   80   90
[2,]  158  184  210
[3,]  246  288  330
```

```
> W[c(1,3),2:1]
     [,1] [,2]
[1,]   80   70
[2,]  288  246
```


```
> mydata <- data.frame(weight=rnorm(5, mean=70, sd=2),
+                      height=rnorm(5, 180, 5))
> mydata
    weight   height
1 71.14088 177.9164
2 74.33516 182.8633
3 70.31349 183.4776
4 68.80590 179.6059
5 71.93287 189.5252
```

```
> str(mydata)
'data.frame':        5 obs. of  2 variables:
 $ weight: num  71.1 74.3 70.3 68.8 71.9
 $ height: num  178 183 183 180 190
> mydata$height
[1] 177.9164 182.8633 183.4776 179.6059 189.5252
> attributes(mydata)
$names
[1] "weight" "height"

$row.names
[1] 1 2 3 4 5

$class
[1] "data.frame"
```


```
> mystring <- c('hello', 'world')
> mylist <- list(list1=mystring, list2=x, list3=W, list4=mydata)
> mylist
$list1
[1] "hello" "world"

$list2
[1]     1     3     9 49049    23

$list3
     [,1] [,2] [,3]
[1,]   70   80   90
[2,]  158  184  210
[3,]  246  288  330

$list4
    weight   height
1 71.14088 177.9164
2 74.33516 182.8633
3 70.31349 183.4776
4 68.80590 179.6059
5 71.93287 189.5252
```

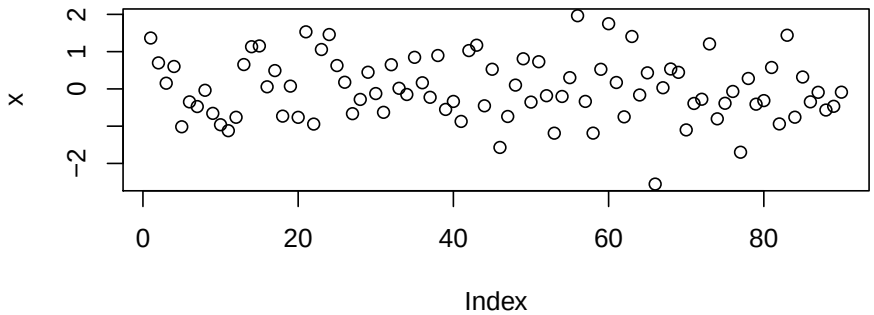

```
> str(mylist)

List of 4
 $ list1: chr [1:2] "hello" "world"
 $ list2: num [1:5] 1 3 9 49049 23
 $ list3: num [1:3, 1:3] 70 158 246 80 184 288 90 210 330
 $ list4:'data.frame':        5 obs. of  2 variables:
  ..$ weight: num [1:5] 71.1 74.3 70.3 68.8 71.9
  ..$ height: num [1:5] 178 183 183 180 190

> attributes(mylist)

$names
[1] "list1" "list2" "list3" "list4"

> names(mylist)

[1] "list1" "list2" "list3" "list4"
```

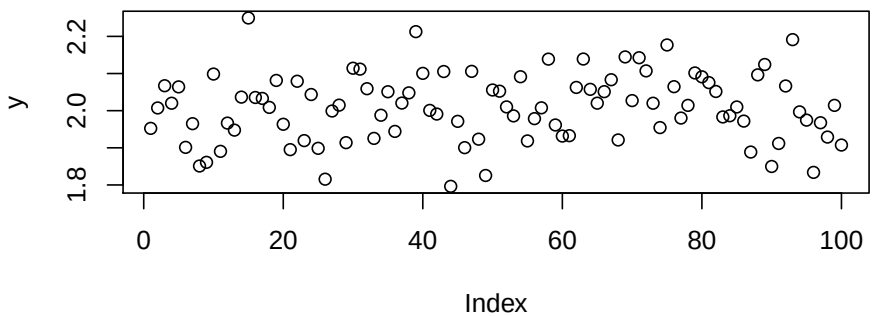
```
> mylist[4]
$list4
    weight   height
1 71.14088 177.9164
2 74.33516 182.8633
3 70.31349 183.4776
4 68.80590 179.6059
5 71.93287 189.5252
> mylist[[4]]
    weight   height
1 71.14088 177.9164
2 74.33516 182.8633
3 70.31349 183.4776
4 68.80590 179.6059
5 71.93287 189.5252
```

```
> mylist$list4
    weight   height
1 71.14088 177.9164
2 74.33516 182.8633
3 70.31349 183.4776
4 68.80590 179.6059
5 71.93287 189.5252
```

- Objects belongs to at least one *class*.
- A *class* can be defined as a family of objects sharing particular features.
- The class of an object determines how it will be treated by *generic* functions.
- A generic function performs a task or action on its arguments specific to the class of the argument itself
- A *method* for a specifc class is a modified version of a generic function tailored for that class
- If the argument lacks any class attribute, or has a class not catered for specifically by the generic function in question, there is always a default action provided.

```
> x <- rnorm(90)
> y <- rnorm(100, 2, .1)
> layout(1:2)
> plot(x)
> plot(y)
> class(x)
[1] "numeric"
> class(y)
[1] "numeric"
```

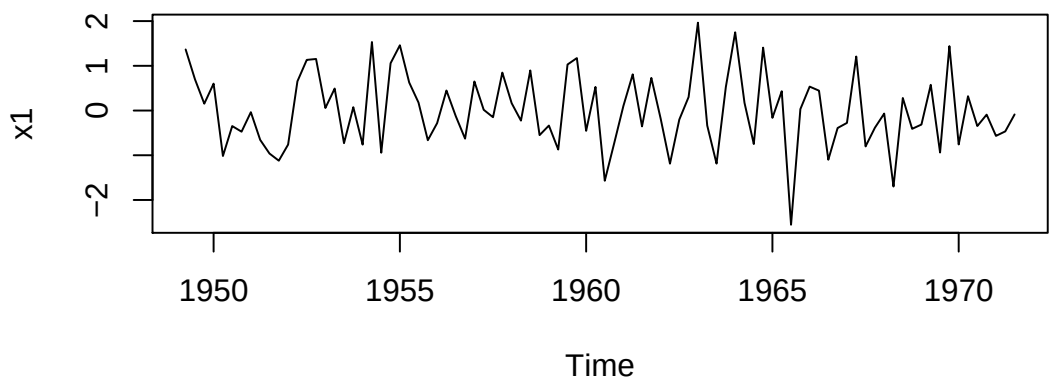


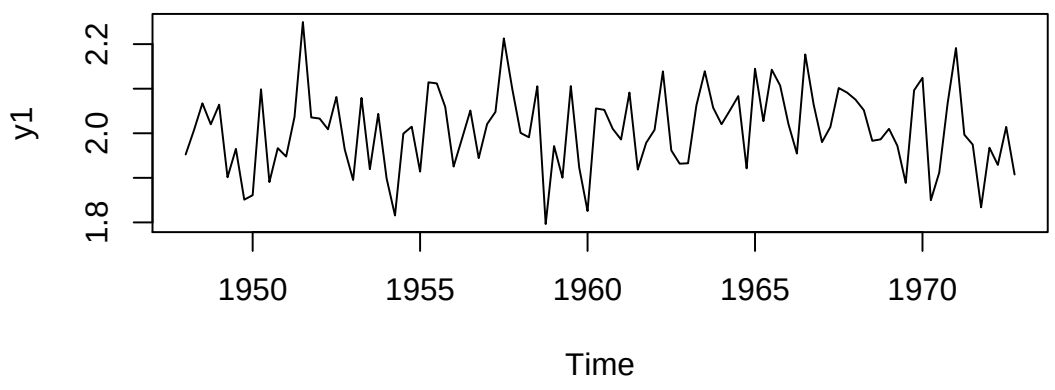


```
> x1 <- ts(data=x, start=c(1949,2), frequency=4)
> y1 <- ts(y, start=c(1948,1), frequency=4)
> class(x1)
[1] "ts"
```


```
> attributes(x1)
$tsp
[1] 1949.25 1971.50    4.00

$class
[1] "ts"
> # tsp provides start date, end date, and frequency
> tsp(x1)
[1] 1949.25 1971.50    4.00
```

```
> window(x1, start=c(1953,3), # extract a subset of the time
+        end=c(1955,4))       # series
           Qtr1        Qtr2        Qtr3        Qtr4
1953                          -0.73006787  0.07465028
1954 -0.76209870  1.53221352 -0.94313012  1.05752980
1955  1.45927153  0.62678784  0.17887615 -0.66386397
> layout(1:2)
> plot(x1)  # actually use the method plot.ts()
> plot(y1)
```


x1
2
1
0
−2
1950
1955
1960
1965
1970
Time

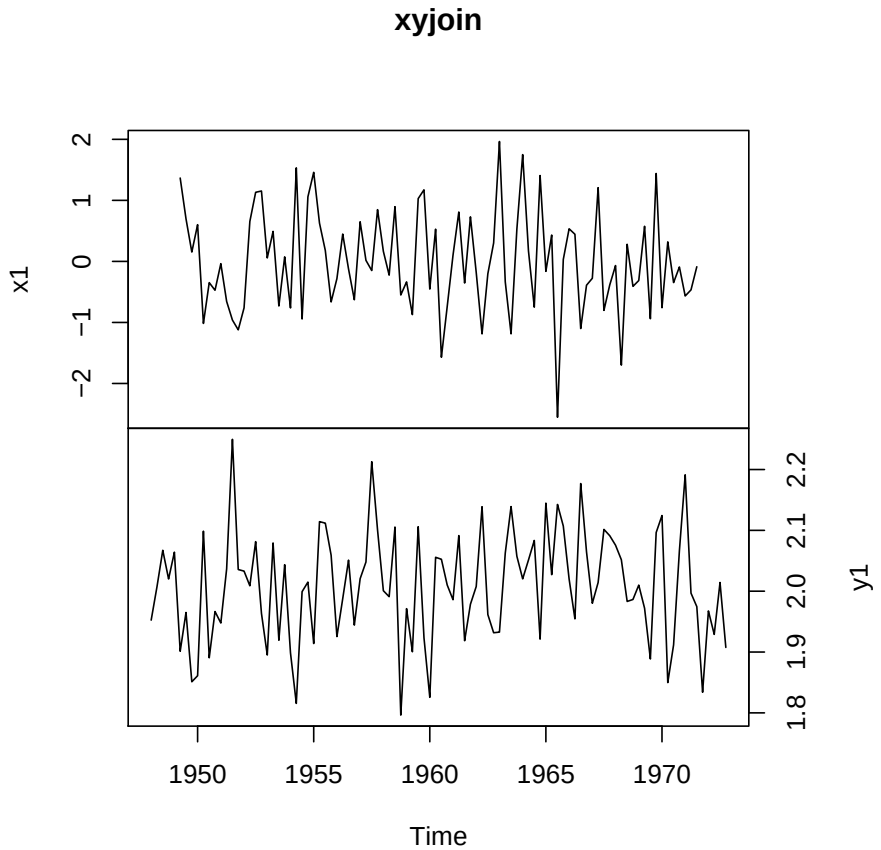

y1
2.2
2.0
1.8
1950
1955
1960
1965
1970
Time


```
> xyjoin <- ts.union(x1,y1)
> window(xyjoin, start=1948.00, end=1950.75)
                x1       y1
1948 Q1         NA 1.952413
1948 Q2         NA 2.007390
1948 Q3         NA 2.067230
1948 Q4         NA 2.020081
1949 Q1         NA 2.064128
1949 Q2  1.3659514 1.901289
1949 Q3  0.6979831 1.965030
1949 Q4  0.1533888 1.851098
1950 Q1  0.6017844 1.861038
1950 Q2 -1.0159784 2.098639
1950 Q3 -0.3472217 1.890536
1950 Q4 -0.4729278 1.966639
```

```
> window(xyjoin, start=1971.00, end=1972.75)
                  x1       y1
1971 Q1 -0.56675915 2.191282
1971 Q2 -0.46636098 1.996790
1971 Q3 -0.08746987 1.974613
1971 Q4          NA 1.833919
1972 Q1          NA 1.967557
1972 Q2          NA 1.928992
1972 Q3          NA 2.014324
1972 Q4          NA 1.907535
```

```
> class(xyjoin)
[1] "mts"    "ts"     "matrix"
> str(xyjoin)
 mts [1:100, 1:2] NA NA NA NA NA ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "x1" "y1"
 - attr(*, "tsp")= num [1:3] 1948 1973 4
 - attr(*, "class")= chr [1:3] "mts" "ts" "matrix"
> tsp(xyjoin)
[1] 1948.00 1972.75    4.00
```

```
> plot(xyjoin, yax.flip=T)
```

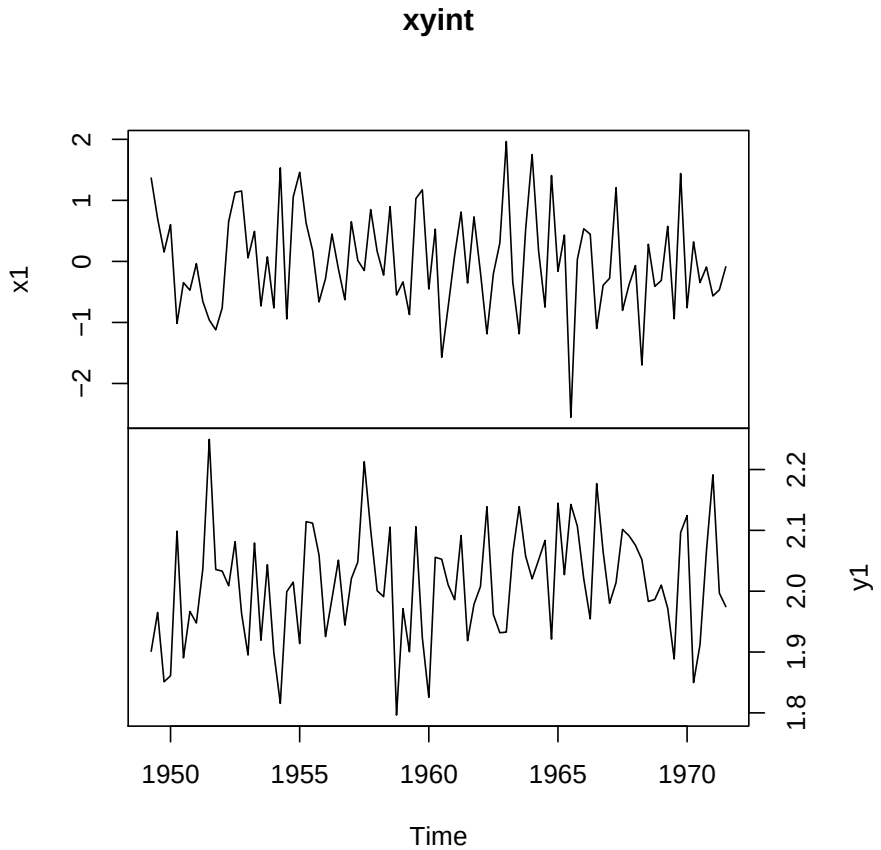



```
> xyint <- ts.intersect(x1,y1)
> tsp(xyint)
[1] 1949.25 1971.50    4.00
> window(xyint, start=1949.25, end=1951.25)
                 x1       y1
1949 Q2  1.36595137 1.901289
1949 Q3  0.69798314 1.965030
1949 Q4  0.15338875 1.851098
1950 Q1  0.60178442 1.861038
1950 Q2 -1.01597843 2.098639
1950 Q3 -0.34722173 1.890536
1950 Q4 -0.47292781 1.966639
1951 Q1 -0.03678975 1.947710
1951 Q2 -0.65585690 2.036590
```

```
> window(xyint, start=1970.00, end=1971.50)
                    x1        y1
1970 Q1 -0.75866027 2.124369
1970 Q2  0.31982246 1.849808
1970 Q3 -0.34695130 1.911767
1970 Q4 -0.09120044 2.066487
1971 Q1 -0.56675915 2.191282
1971 Q2 -0.46636098 1.996790
1971 Q3 -0.08746987 1.974613
```

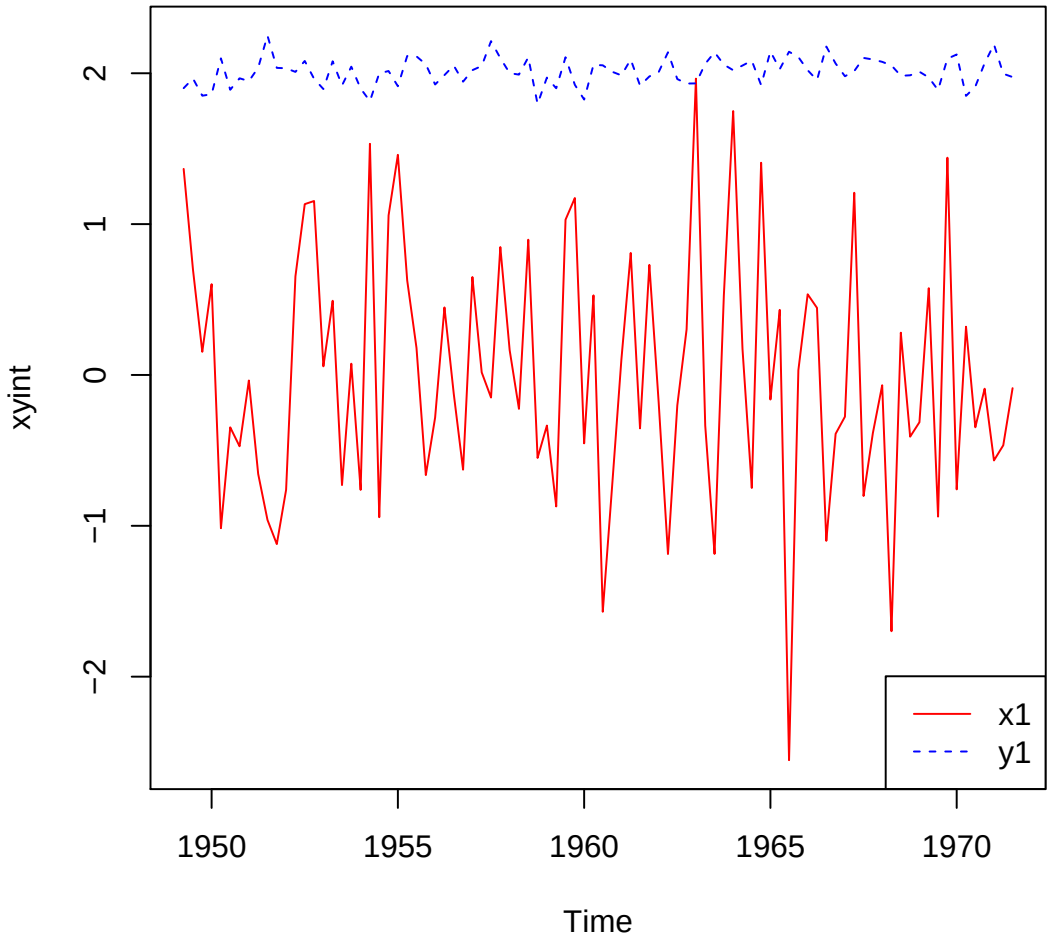
```
> class(xyint)
[1] "mts"    "ts"     "matrix"
> str(xyint)
 mts [1:90, 1:2] 1.366 0.698 0.153 0.602 -1.016 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr [1:2] "x1" "y1"
 - attr(*, "tsp")= num [1:3] 1949 1972 4
 - attr(*, "class")= chr [1:3] "mts" "ts" "matrix"
> tsp(xyint)
[1] 1949.25 1971.50    4.00
```

```
> plot(xyint, yax.flip=T)
```



A different plot

```
> plot(xyint, plot.type='single',
+      col=c('red','blue'), lty=1:2)
> legend(x='bottomright', legend=c('x1', 'y1'),
+        col=c('red', 'blue'), lty=1:2)
```


xyint
2
1
0
−1
−2
1950
1955
1960
1965
1970
Time
x1
y1

- A *package* is a bunch of functions (usually defined for specific problems).
- Standard (or *base*) packages are part of R source code.
- Contributed packages can be downloaded and installed: there are thousands of contributed packages!
- After installation, a contributed package can be loaded by one of the following commands: `library(packagename)`, or `require(packagename)`.
- During the course we shall use a few packages well suited for time series analysis.