|  |
| --- |
| National Bank of Belgium |
| JD+ |
| Call from R |
|  |
| **Palate Jean** |
| **1/27/2015** |

## Basic explanations

JD+, like any other java library, can be called from the software “R” through de R-package “rJava”. See for instance <http://cran.r-project.org/web/packages/rJava/index.html>

The design of JD+ - at least of most of its high-level routines – greatly simplifies the interaction between JD+ and R. More precisely, the input/output of those routines can be made by a small number of basic concepts (for instance time series, arrays of doubles…), which can be re-presented by some R-structures. The R-routines that translates those R-structures from/to Java structures are implemented in a few R-script files, listed below:

|  |  |
| --- | --- |
| File | Contents |
| jd\_init.R | Basic structures (enum and SA factories) |
| jd\_ts.R | Time series and related concepts ( periods…) |
| jd\_sa.R | Main SA routines (Tramo-Seats and X13 |
| jd\_rslts.R | Reading of the results from a high-level algorithm |
| jd\_spec.R | Creation of the specification for a high-level algorithm |

|  |  |
| --- | --- |
| Java concept | R concept |
| ec.tstoolkit.timeseries.TsFrequency  and other enums | integer |
| ec.tstoolkit.timeseries.TsPeriod | array (freq, year, period+1) |
| ec.tstoolkit.timeseries.TsData | ts structure |
| ec.tstoolkit.Parameter | array (value, stde) |
| ec.tstoolkit.information.StatisticalTest | array with annotation (value, pvalue + distribution) |
| ec.tstoolkit.information.Regression item | array with annotation (value, stde + description) |
| ec.tstoolkit.math.Matrix | matrix |
|  |  |

All the initialization is processed by the following code

|  |
| --- |
| source("…/jd\_init.R")  source("…/jd\_ts.R")  source("…/jd\_sa.R")  source("…/jd\_rslts.R")  source(".../jd\_spec.R") |

## Example

The example below can be found in the R-script “jd\_test.R”

### Basic execution of Tramo-Seats and of X13

The execution of Tramo-Seats or of X13 using pre-specified specifications is straightforward:

Call the function “sa\_xx” with an R-time series and the name of the specification and store the results in a R-wrapper around the Java output.

|  |
| --- |
| # usual R time series  data<-read.table("./xm.txt")  s<-ts(data[,1], start=c(1995,1), frequency=12)  # executes TramoSeats (RSAfull by default)  tramoseats\_rslts=sa\_tramoseats(s)  # executes X11/X13 (pre-defined specifications)  x11\_rslts<-sa\_x13(s, "X11")  x13\_rslts<-sa\_x13(s, "RSA5c") |

### Information retrieval

The information retrieval is performed by means of specialized functions, which are able to translate Java structure in R-structure. The identification of the information is performed by means of the JD+ dictionary for the output. It is exactly the same dictionary as the dictionary used for generating the outputs in the graphical interface of JD+ or of the cruncher. See the document JD+\_Output.docx for more information

|  |
| --- |
| # retrieve the seasonally adjusted series  sa0<-proc\_ts(x13\_rslts, "sa")  sa1<-proc\_ts(tramoseats\_rslts, "sa")  #trend  t0<-proc\_ts(x13\_rslts, "t")  t1<-proc\_ts(tramoseats\_rslts, "t")  #d7 table  d7<-proc\_ts(x13\_rslts, "decomposition.d-tables.d7")  #series corrected for calendar effects  ycal0<-proc\_ts(x13\_rslts, "ycal")  ycal1<-proc\_ts(tramoseats\_rslts, "ycal") |

Further examples with the corresponding output (in blue) are provided below

|  |
| --- |
| > # regression variables  > proc\_desc(x13\_rslts, "regression.description")  [1] "Week days" "Easter [1]" "AO (6-2000)" "AO (12-2005)" "AO (11-2008)"  >  > #regression coefficients. Value/standard deviation. See description for their meaning  > proc\_parameters(x13\_rslts, "regression.coefficients")  [,1] [,2]  [1,] 0.009798992 0.0005094853  [2,] -0.032404300 0.0072503198  [3,] -0.106843550 0.0210762280  [4,] 0.088772602 0.0212856714  [5,] -0.115648205 0.0289050436  >  > #test Value/PValue  > proc\_test(x13\_rslts, "residuals.lb")  [1] 13.5773346 0.9157994  attr(,"description")  [1] "Chi2 with 22 degrees of freedom "  > proc\_test(x13\_rslts, "residuals.skewness")  [1] 0.08346501 0.67240269  attr(,"description")  [1] "Normal with Mean = 0.0 and Stdev = 0.19738550848793068"  >  > #BIC  > proc\_numeric(tramoseats\_rslts, "likelihood.bicc")  [1] -6.891744  > proc\_numeric(tramoseats\_rslts2, "likelihood.bicc")  [1] -6.828965  > |

### Advanced use: defining a new specification

The user is able to generate new specification that will be used with the main SA routines. The new specification will overwrite some of the options of a pre-specified specification.

The way of initializing a specification object is parallel to the way information is retrieved from an output object. The user has to set the right type of parameter with the corresponding identifier, which can be found in the dictionary of the algorithm. See the document “InputDictionaries.docx” for the possible codes.

|  |
| --- |
| # advanced processing  # create a spec file that will modify an existing specification  spec<-spec\_create()  spec\_bool(spec, "tramo.automdl.enabled", FALSE)  spec\_fixedparams(spec, "tramo.arima.btheta", -.8)  spec\_nparams(spec, "tramo.arima.phi", 2)  spec\_bool(spec, "tramo.regression.calendar.td.auto", TRUE)  # execute TramoSeats on the series s, using the "RSA4" specification modified by the given spec details (see above)  tramoseats\_rslts2=sa\_tramoseats(s,"RSA4",spec) |

## Final remarks

The link between R and JD+ uses to a large extent the same mechanism as the input/output channels of the graphical interface and of the cruncher. It has been found that it is very efficient.

The link is not complete yet. For instance, the use of user-defined calendars and regression variables must still be developed.

As the same I/O mechanism is used for all the high-level routines of JD+, the solution used for seasonal adjustment could be extending in a straightforward way to other algorithms (benchmarking…).