|  |
| --- |
| National Bank of Belgium |
| JD+ |
| Call from R |
|  |
| **Palate Jean** |
| **3/2/2016** |

## 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 |
| jd\_calendars.R | Concepts related to calendars |
| jd\_regression.R | Concepts related to regression variables |
| jd\_cholette.R | Basic routines for benchmarking (Cholette/Denton) |
| jd\_tempdisagg.R | Basic routines for temporal disaggregation |

|  |  |
| --- | --- |
| 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")  source("./jd\_regression.R")  source("./jd\_calendars.R")  source("./jd\_cholette.R")  source("./jd\_tempdisagg.R") | Basic objects  Basic objects  Seasonal adjustment  Information set output  Information set input  Regression variables  Calendars  Benchmarking  Temporal disaggregation |

Examples and tests can be found in the following files

|  |  |
| --- | --- |
| Seasonal adjustment | ./jd\_testsa.R |
| Calendars | ./jd\_testcalendars |
| Temporal disaggregation | ./jd\_testdisagg (needs also the R-package tempdisagg) |

## SA routines

The example below can be found in the R-script “”

### 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) |

## Benchmarking and temporal disaggregation

### jd\_cholette

#### Description

Benchmarking by means of the Cholette’s method

#### Usage

jd\_cholette(s, t, rho=1, lambda=1, conversion=”Average”)

#### Arguments

|  |  |
| --- | --- |
| s | source = original series, which must be benchmarked. |
| t | target = aggregation constraint. |
| rho | auto-regressive parameter. |
| lambda | power of the weights applied to the series. 0 for additive, 0.5 for proportional, 1 for multiplicative. |
| conversion | Sum, Average, [First, Last]. |

#### Output

R time series

#### Source

jd\_cholette.R

### jd\_denton

#### Description

Benchmarking by means of the Denton’s method

#### Usage

jd\_denton(s, t, mul=TRUE, modified=TRUE, d=1, conversion="Average")

#### Arguments

|  |  |
| --- | --- |
| s | Source = original series, which must be benchmarked. |
| t | Target = aggregation constraint. |
| mult | Multiplicative or additive benchmarking. |
| modifed | Use of the modification Denton procedure, which affects the initial values. Should be usually set to TRUE |
| d | Power of the differences used in the objective function |
| conversion | Sum, Average, [First, Last]. |

#### Output

R time series

#### Source

jd\_cholette.R

### jd\_td

#### Description

Temporal disaggregation of a time series. This function is largely inspired by the function “td” of the package “tempDisagg”. However, it uses a completely different implementation (based on a Kalman smoother)

#### Usage

jd\_td (formula, model="Ar1", conversion="Sum", zeroinit=FALSE, truncated.rho=0, fixed.rho=-1, to=4)

#### Arguments

|  |  |
| --- | --- |
| formula | Description of the regression model. The usual R formulae are supported. See details. |
| model | model of the residuals. Acceptable values are:   |  |  | | --- | --- | | Wn |  | | Ar1 |  | | Rw |  | | RwAr1 |  | | I2 |  | | I3 |  | |
|  |  |
| conversion | Sum, Average, [First, Last]. |
| mult | Multiplicative or additive benchmarking. |
| zeroinit | Zero initialization of the error term |
| truncated.rho | limit for the search of . Only used with Ar1 and RwAr1. |
| fixed.rho | Fixed value for . Should be in ]-1,1[. Only used with Ar1 and RwAr1. When this value is set to -1, is estimated by maximum likelihood. |
| to | Annual frequency of the disaggregated series. Used only when the model as no time series in its explanatory variables. |

#### Output

R time series

#### Source

jd\_tempdisagg.R

#### Details

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Formula | Model | Others |
| Chow-Lin with intercept | y ~ q (+ 1) or y ~ q | Ar1 | truncated.rho=0 |
| Chow-Lin without intercept | y ~ q + 0 | Ar1 | truncated.rho=0 |
| Fernandez | y~q + 0 or  y~q (+1) | Rw  Rw | zeroinit=TRUE |
| Litterman | y~q + 0 or  y~q (+1) | RwAr1  RwAr1 | truncated.rho=-1  zeroinit=TRUE, truncated.rho=-1 |

Remark:

The current code should be improved to generate more results (coefficients…)

## Calendars

It is possible to generate and to manage JD+ calendar in R.

The main routines are descrived in the following example (jd\_tescalendars.R)

|  |
| --- |
| # Create a new calendar  jdc<-jd\_calendar()  # Add fixed days  # 1 May (month/day)  jd\_addFixedDay(jdc, 5, 1)  # 21 July  jd\_addFixedDay(jdc, 7, 21)  # Add Easter related days  #Ascension (=Easter+39)  jd\_addEasterRelatedDay(jdc, 39)  #White Monday (=Easter+50)  jd\_addEasterRelatedDay(jdc, 50)  # Unregister / register the calendar  jd\_unregistercalendar("mycalendar")  jd\_registercalendar("mycalendar", jdc)  # Create a time domain  jd\_dom<-domain\_r2jd(c(12,1980,1, 120))  # Generate the time matrix corresponding to the calendar and to a given domain  # Computes “TradingDays” or “WorkingDays”  jd\_calendarData(jdc, c(12,1980,1, 28\*12))  # Retrieve a registered calendar  jdc2<-jd\_getcalendar("mycalendar")  jd\_calendarData(jdc2, c(4,1980,1, 28\*4), "wd")  #using a calendar in seasonal adjustment  # usual R time series  data<-read.table("../Data/xm.txt")  s<-ts(data[,1], start=c(1995,1), frequency=12)  r<-sa\_tramoseats(s, "RSAfull")  proc\_ts(r, "sa")  spec<-spec\_create()  spec\_str(spec,"tramo.regression.calendar.td.holidays" ,"mycalendar")  spec\_str(spec,"tramo.regression.calendar.td.holidays" ,"mycalendar")  rh<-sa\_tramoseats(s, "RSAfull", spec)  proc\_ts(rh, "sa")  ts.plot(proc\_ts(r, "tde"), proc\_ts(rh, "tde"), col=c("black", "red")) |