Introducing the potential of rjd3sts

Workshop on Time Series Analysis and Statistical Disclosure Control Methods for Official Statistics

December 14-15, 2023

Corentin Lemasson | OECD Paris

BanqueNationaleBank DE BELGIQUE VAN BELGIË
Eurosystem

Outline

1. rjd3sts in a nutshell
2. State Space Framework
3. Added value of *rjd3sts* compared with other JDemetra+ tools
4. Case studies
   - Seasonal adjustment of series that have changed frequency
   - Mixed-frequency imputation tool
5. Links

rjd3sts in a nutshell

- R package of JDemetra+ v3 dedicated to Structural models and State Space models
- Main features
  - Rich State Space tool
    - Large collection of blocks, uni/multivariate, bunch of output…
  - High-performance
    - Interface to JAVA libraries (JDemetra+ v3)
    - Fast processing: functional forms instead of matrix computations
    - Parameters estimated by ML through specialized optimization procedures
  - Easy to use
    - Step-by-step approach and few technical details in R

State Space Framework

- A **unified approach** to a wide number of problems in time series
- Consists of the construction of dynamic models, with **Markovian process**
  - i.e., state in T+1 only dependent on the state in T
  - Cover traditional models and allows us to go further
- **Blocks**-based approach
  - Complex models can thus be simply built

State Space Framework (2)

- State composed of independent blocks

$$\begin{bmatrix} \alpha_{1,t+1} \\ \alpha_{2,t+1} \\ \vdots \\ \alpha_{q,t+1} \end{bmatrix} = \begin{bmatrix} T_{1,t} & 0 & \cdots & 0 \\ 0 & T_{2,t} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & T_{q,t} \end{bmatrix} \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \\ \vdots \\ \alpha_{q,t} \end{bmatrix} + \mu_t, \qquad \mu_t \sim N(0, \begin{bmatrix} \Omega_{1,t} & 0 & \cdots & 0 \\ 0 & \Omega_{2,t} & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \Omega_{q,t} \end{bmatrix})$$

- Measurement equation links the state to the (multivariate) observations

$$y_{i,t} = [Z_{1,i,t} \quad Z_{2,i,t} \quad \cdots \quad Z_{q,i,t}] \begin{bmatrix} \alpha_{1,t} \\ \alpha_{2,t} \\ \vdots \\ \alpha_{q,t} \end{bmatrix} + \varepsilon_{i,t},$$

Current catalog of rjd3sts

- “Atomic” blocks with automatic (diffuse) initialization and dynamics
  - ARIMA, local level/linear trend, seasonal components, (time varying) regression coefficients, VAR…
- Derived blocks
  - Aggregation of the atomic blocks, cumulator (handling of mixed frequencies)
- Large collection of default or specific measurement blocks
  - Selectors, regression variables…

Added value of rjd3sts compared with other JDemetra+ tools

- Allow to solve many problems in time series, including things that cannot be done with other tools

- Examples (see https://github.com/rjdemetra/rjd3sts/wiki)
  - Mixed frequency → e.g. see following case studies
  - Time varying-coefficient (e.g. in calendar effect)
  - Break in some components of the series, but not in the others (e.g. seasonal break, while the trend is preserved)
  - Heteroskedasticity (e.g. during Covid time)
  - Adding constraints (e.g. fixing values at some periods of time)
  - Estimate of the precision of the various components
  - …

- Suggestion: use *rjd3sts* instead of traditional tools for problematic cases

Case studies

- Mixed frequency cases
  1. Seasonal adjustment of series that have changed frequency
  2. Mixed-frequency imputation tool
- Both requires the use of a "cumulator[1]" block

Initial state block

$$M = \begin{cases} \alpha_t, & state \\ T_t, & transition \\ Z_t, & loading \\ & \dots \end{cases}$$

Derived state block

$$M^c = \begin{cases} \tilde{\alpha}_t = \begin{pmatrix} \sum_{j=t0}^{t-1} (Z_j \alpha_j) \\ \alpha_t \end{pmatrix} \\ \tilde{T}_t = \begin{pmatrix} 1 & Z_t \\ 0 & T_t \end{pmatrix} \\ \tilde{Z}_t = \begin{pmatrix} 1 & Z_t \end{pmatrix} \\ \dots \end{cases}$$

[1]Proietti (2005): Temporal Disaggregation by State Space Methods: Dynamic Regression Methods Revisited

1. Seasonal adjustment of series that have changed frequency

Bookstores retail series

Input

	Monthly part of the series	Quarterly part of the series	Cumulated series by quarter
01/01/1992	790		790
01/02/1992	540	NA	1330
01/03/1992	536		1866
01/04/1992	524		524
01/05/1992	553	NA	1077
01/06/1992	589		1666
…	…	…	…
01/10/2005	1064		1064
01/11/2005	1150	NA	2214
01/12/2005	2256		4470
01/01/2006	NA		NA
01/02/2006	NA	4418	NA
01/03/2006	NA		4418
01/04/2006	NA		NA
01/05/2006	NA	3401	NA
01/06/2006	NA		3401
…	…	…	…
01/10/2010	NA		NA
01/11/2010	NA	3961	NA
01/12/2010	NA		3961

R Code

```
library(rjd3toolkit)
library(rjd3sts)

# Import and format input (sc)
# ...

# Define your calendar regressors using functions from rjd3toolkit
# ...

# Set up the model (state)
bsm<-model()

# Define components, i.e. the 'atomic' blocks
trend<-locallineartrend("trend")
seas<-seasonal("seas", frequency(s), type="HarrisonStevens")
cal<-reg("regcal", regcal)
noise<-noise("noise")

# Aggregate the components and derive the cumulator block
all<-aggregation("m", list(trend, seas, cal, noise))
c<-cumul("c", all, period = frequency(sm)/frequency(sq))

# Add the derived cumulator to the state
add(bsm, c)

# In the last version of rjd3sts, the measurement equation is built
# automatically with default loadings when not defined like here

# Estimate the model
rslt<-estimate(model=bsm, data=sc)
```

Output

> dictionary(rslt)

```
 [1] "likelihood.ll"            "likelihood.ser"           "likelihood.residuals"    "scalingfactor"           "ssf.ncmps"
 [6] "ssf.cmppos"               "ssf.cmpnames"             "parameters"              "parametersnames"         "fn.parameters"
[11] "ssf.T(*)"                 "ssf.V(*)"                 "ssf.Z(*)"                "ssf.P0"                  "ssf.B0"
[16] "ssf.smoothing.array(?)"   "ssf.smoothing.varray(?)"  "ssf.smoothing.cmp(?)"    "ssf.smoothing.vcmp(?)"   "ssf.smoothing.state(?)"
[21] "ssf.smoothing.vstate(?)"  "ssf.smoothing.states"     "ssf.smoothing.vstates"   "ssf.filtering.array(?)"  "ssf.filtering.varray(?)
[26] "ssf.filtering.cmp(?)"     "ssf.filtering.vcmp(?)"    "ssf.filtering.state(?)"  "ssf.filtering.states"    "ssf.filtering.vstates"
[31] "ssf.filtering.vstate(?)"  "ssf.filtered.array(?)"    "ssf.filtered.varray(?)"  "ssf.filtered.cmp(?)"     "ssf.filtered.vcmp(?)"
[36] "ssf.filtered.state(?)"    "ssf.filtered.vstate(?)"   "ssf.filtered.states"     "ssf.filtered.vstates"
```

> result(rslt, "ssf.smoothing.states")

	[,1]	[,2]	[,3]	[,4]	[,5]	…	[,13]	[,14]	[,15]	[,16]	…	[,20]	[,21]	[,22]
[1,]	0.0	661.9	6.1	106.1	461.1	…	-179.5	-97.9	8.0	-15.6	…	-2.3	34.0	-9.74E-09
[2,]	790.0	669.7	6.1	-127.4	120.0	…	-131.8	-180.7	8.0	-15.6	…	-2.3	34.0	2.50E-09
[3,]	1330.0	677.1	6.1	-100.1	-132.3	…	-99.5	-132.7	8.0	-15.6	…	-2.3	34.0	2.07E-09
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

> regcal

	Mon	Tue	Wed	Thu	Fri	Sat	easter
Jan-92	0	0	0	1	1	0	0
Feb-92	0	0	0	0	0	1	0
Mar-92	0.40636	0.20318	-0.7968	-0.7968	-0.7968	-0.7968	-0.5
…	…	…	…	…	…	…	…

→ Series

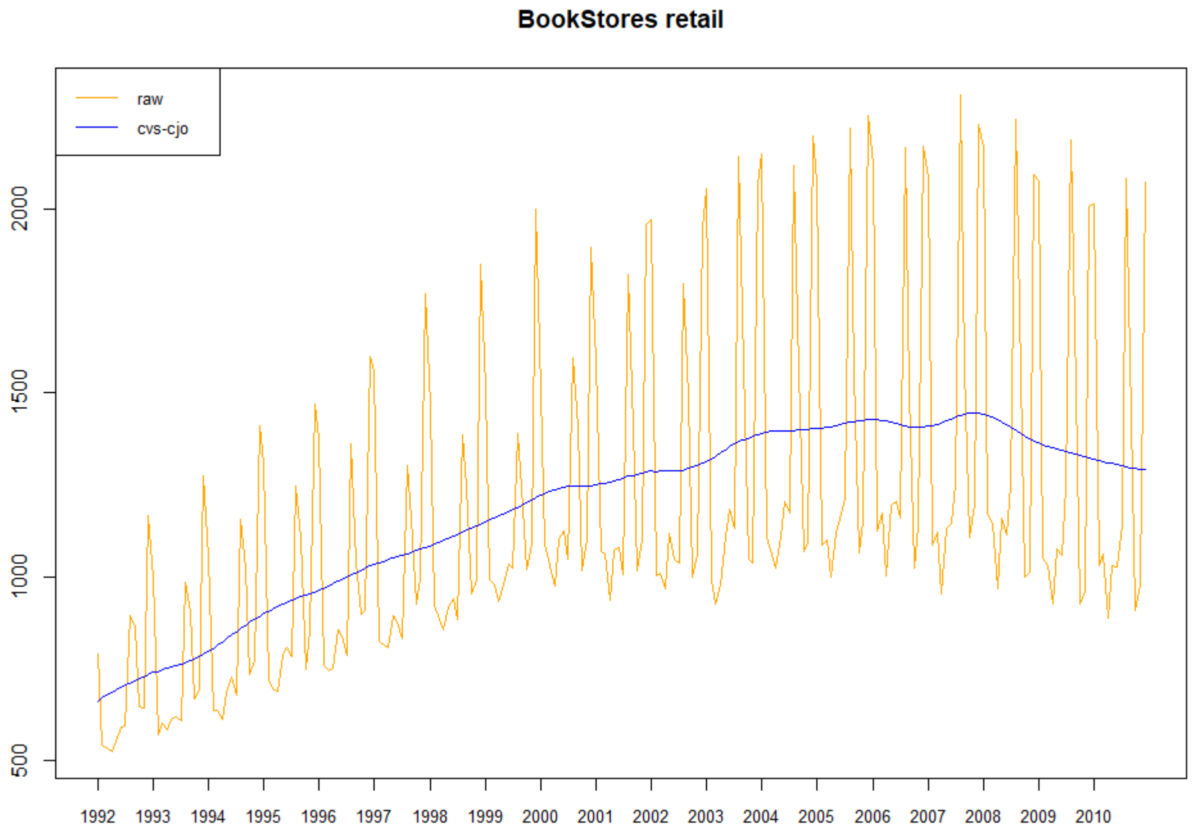
sa	[,2]+[,22]
raw	[,2]+[,4]+sumproduct([,15]:[,21],regcal)+[,22]

> result(rslt, "ssf.smoothing.vstates")

…

R Code

```
ss<-result(rslt, "ssf.smoothing.states")
s_raw<-ss[,2]+ss[,4]+rowSums(ss[,15:21]*regcal)+ss[,22]
s_sa<-ss[,2]+ss[,22] # sa = trend + noise
plot(s_raw, type='l', xaxt="n", col='orange', ylab="", xlab="", main="BookStores retail")
lines(s_sa, col='blue')
axis(1, seq(1,length(s),12), 1992:2010, cex.axis = .8)
legend("topleft", legend=c("raw", "cvs-cjo"),col=c("orange", "blue"), lty=1, cex=0.8)
```



2. Mixed-frequency imputation tool

- Context
  - Quarterly estimates of government accounts suffer from inconsistencies and delays in the delivery of source data
  - However, fairly accurate estimate from the annual budget is available for the current year
  - Question: how to use this information to improve the estimates of the missing quarters?
- Solution
  - Use of a structural model with cumulator
- Potential difficulties for the end user
  - What model?, outliers?, calendar effects? what's in my state? How to extract the results I need?...
  - → Add an extra layer: e.g. package R *nbbSTSestimate* (see https://github.com/clemasso/nbbSTSestimate)

nbbSTSestimate

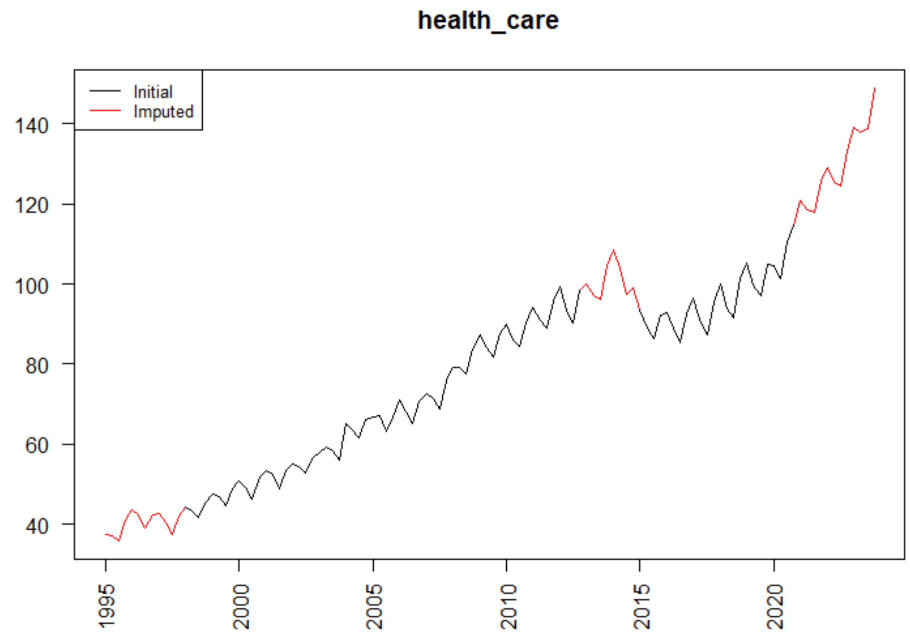
- Current features
  - Allow monthly, quarterly or annual data with or without low-frequency input
  - Single or multi-processing
  - Common structural models are included with the possibility to add outliers and calendar effects
  - Automatic modelling/detection
  - Input checks
  - Output quite rich

R Code

```r
library(nbbSTSestimate)

# Import input (sq and sy)
# ...

rslt<-estimateSTS(sq, sy, stsmodel="auto", outliers="auto", cal.effect="auto", cal.effect.td = "BE")
# or rslt<-estimateSTS_fromXLSX("input.xlsx", is.lf = TRUE)

rslt$series$table
rslt$series$regressors$param
plot(rslt)
#...
```

```
> rslt$health_care$table
          Series    Trend      Slope    Seasonal   Irregular
1995 Q1 37.61112 35.57290 1.69473339 2.06165700 -0.023441002
1995 Q2 37.31840 37.26764 1.63944944 0.07420632 -0.023441002
1995 Q3 35.98202 38.90709 1.47359762 -2.90162637 -0.023441002
1995 Q4 41.11999 40.38068 1.14189397 0.76275038 -0.023441002
1996 Q1 43.65863 41.52258 0.58905455 2.06366544 0.072384698
1996 Q2 42.26028 42.11163 -0.01420645 0.07625893 0.072384698
1996 Q3 39.27024 42.09743 -0.49717486 -2.89957376 0.072384698
1996 Q4 42.44159 41.60025 -0.68913648 0.76895236 0.072384698
1997 Q1 42.87910 40.91111 -0.41937714 2.05746346 -0.089473225
1997 Q2 40.47506 40.49174 0.10108690 0.07279190 -0.089473225
1997 Q3 37.60031 40.59282 0.66123937 -2.90304079 -0.089473225
1997 Q4 41.92588 41.25406 1.05006402 0.76128622 -0.089473225
1998 Q1 44.48696 42.30413 1.05654457 2.06512960 0.117700748
1998 Q2 43.39747 43.36067 0.95826990 0.07722961 -0.040431526
1998 Q3 41.66977 44.31894 0.65988513 -2.90199031 0.252820327
1998 Q4 45.61177 44.97883 0.75764908 0.77426521 -0.141322368
1999 Q1 47.64686 45.73648 0.91826296 2.03933488 -0.128948783
```

```
> rslt$health_care$regressors$param
    AO.36     LS.103
-6.946467 11.588629
```



Links to packages

- rjd3sts: https://github.com/rjdemetra/rjd3sts
- nbbSTSestimate: https://github.com/clemasso/nbbSTSestimate