TSACE Webinar, Wednesday December 14th 2022

Using JDemetra+ in R: from version 2 to version 3
Presentation 4: SA production and qulaity report in R

Anna Smyk and Tanguy Barthelemy
With the collaboration of Alain Quartier-la-tente

Contents

Tackling Production issues

massive data sets

Contents

Quality report with JDCruncheR (1/3)

JDemetra+ Cruncher (executable module) allows to - update a JDemetra+ workspace (refresh policy) - export the results (series and diagnostics), without having to open the graphical interface and operate manually.

It can be launched in R with rjwsacruncher or JDCrunhcerR packages.

The JDCruncheR package also:

- computes a score (based on “Good”, “Bad” modalities og selected diagnostics)
- creates a quality report from the diagnostics produced by JDemetra+

The three main functions of the package are:

- `extract_QR` to extract the quality report from the csv file (demetra_m.csv) that contains all JD+ diagnostics;
- `compute_score` to compute a weighted score based on the diagnostics
- `export_xlsx` to export the quality report.

Quality report with JDCruncheR (2/3): example

```r
# choose the demetra_m.csv file generated by the cruncher
QR <- extract_QR("../Output/SA")
QR

?compute_score # to see how the score is calculated (formula)
QR <- compute_score(QR,
                    n_contrib_score = 3)


QR

QR <- sort(QR, decreasing = TRUE, sort_variables = "score")
export_xlsx(QR,
            file_name = "U:/quality_report.xls")
```

When working with several workspaces (or SAPs), quality reports can be piled up with the function `rbind()` or by creating a mQR_matrix object with the function `mQR_matrix()`

Quality report with JDCruncher (3/3) : example

Missing values can be ignored and conditions can be set for indicators:

```r
# oos_mse weight reduced to 1 when the other
# indicators are "Bad" ou "Severe"
condition1 <- list(indicator = "oos_mse",
                   conditions = c("residuals_independency",
                                  "residuals_homoskedasticity",
                                  "residuals_normality"),
                   conditions_modalities = c("Bad","Severe"))
BQ <- compute_score(BQ, n_contrib_score = 5,
                    conditional_indicator = list(condition1),
                    na.rm = TRUE)
```

Example of score composition

Diagnostics		Weights (out of 100)
Pre-adjustment	ARIMA Model Residuals	30
	Residual Calendar Effects	20
Decomposition	Residual seasonality	45
	Decomposition Quality (stats M if X11)	5

Customize the score computation

Practical steps if you want to customize the score computation (see package documentation in R)

- select your indicators of interest
- adjust “good”, “bad”... threshold in JD+ GUI if necessary
- by default good=0, uncertain=1, bad or severe=3
- change this grading system and/or the weights directly in the package functions
- rebuild your package

Warning : here only diagnostics are taken into account, revisions and numerical effects of potential parameter tuning have to be analysed with a complementary tool

Contents

SA production (fully ?) in R

A request wich comes back all the time

- flexibility of data format
- feel of better automatization

Here contrast

- “old fashion set-up”: WS created in GUI, readable with GUI refreshable with the cruncher some R might be used for...
- “full R set-up”: no ws structre, time series object

Data format and portability

Tuning specifications

(Process set up or annual review)

Massive data set eache series (or goup of series) has specific (pre-determined) parameters: - pre-specified outliers - calendar regressors

Estimation and Refreshing data

(Annual or infra anual reviews)

from P2 (everything here or split ?)

Annual review

comparing old and new sets of params “current” params vs automiatic reestimation (with some user-def params)

Selective editing and Manual fine tuning

select series

looking ad diagnostics/ fine tuning params (loop)

with or without GUI

reading data, comparing numerical impact of params

Contents

On production in R

Assets of WS-GUI-Cruncher set up (with some R help) GUI for manual fine tuning

Assets of “Full R set up”

Take home message

summary

- what is new
- what is missing

Possible Contributions

- Testing it and reporting issues
- Developping new tools (other packages, new functions, etc.)

Resources

- **Webinar Resources** on GitHub: slides, code, additionnal references (Beamers and papers) https://github.com/annasmyk/Tsace_RJD_Webinar_Dec22
- Coming soon: **JDemetra+ NEW online documentation first release** on Thursday december 22nd: https://jdemetra-new-documentation.netlify.app/

Restriced scope : SA (incl HF) and Chapter on Tools (GUI, R packages and plung-ins)

- Blog **JDemetra+ universe** https://jdemetra-universe-blog.netlify.app/
  - can be used for problem/solution/insights sharing (comments available if logged into GitHub)
  - guest posts welcome
  - will link “all” presentations about JDemetra+ in confs / workshop (so if you give a talk about JD+ let us know..)

Thank you for your attention

Packages R:

- palatej/rjd3toolkit
- palatej/rjd3modelling
- palatej/rjd3sa
- palatej/rjd3arima
- palatej/rjd3x13
- palatej/rjd3tramoseats
- palatej/rjdemetra3
- palatej/rjdfilters
- palatej/rjd3sts
- palatej/rjd3stl
- palatej/rjd3highfreq
- palatej/rjd3bench
- AQLT/ggdemetra3