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

ANNA SMYK AND TANGUY BARTHELEMY
With the collaboration of Alain Quartier-la-tente

Contents

Tackling Production issues

Context of use :

- massive data sets
- setting up a production process
- annual or infra-annual reviews
- producing quality reports
- manual fine tuning a selected sub-set of series

Contents

Quality report with JDCruncheR package (1/4)

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 JDCruncherR packages.

The **JDCruncheR package** produces a QR based on JDemetra+ cruncher output

Quality report with JDCruncheR package (2/4)

The three main functions of the JDCruncher 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 selected diagnostics and corresponding "Good", "Bad",... modalities
- `export_xlsx` to export the quality report to Excel

Quality report with JDCruncheR (3/4): example

```
# 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")
```

Quality report with JDCruncher (4/4) : example

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

```
# 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

Future developments: make this functions directly customizable by the user

In this QR only diagnostics are taken into account, revisions and numerical effects of potential parameter tuning still have to be analysed

Contents

SA production (fully?) in R

A request which comes back all the time

- better (?) automation when the remainder of the production process (outside of SA) is also done in R

We will contrast

- "old fashion set-up": workspace created in GUI, readable with GUI and refreshable with the cruncher (functions from R packages might be used as auxiliary tools, e.g "instant read"..)
- "full R set-up": no workspace structure, time series objects only

Data format and portability

Workspace created in GUI:

- rigid data structure (series order constraints)
- physical path to data (not portable)

Time-series objects in R: complete flexibility and portability (Sharing R projects can be done easily)

Fine-tuning specifications

Context of use :

- SA processing first set-up or annual review
- massive data set
- each series has specific (pre-determined) parameters: pre-specified outliers, calendar regressors

Fine-tuning specifications:

- In a classical Workspace: not easy, need for an auxiliary tool (in java for example)
- In a full R set-up: easier to write code generating specific (different) specifications in large number and link them to series

Estimation and Refreshing data

In a classical workspace :

- very easy and fast with the Cruncher

In a full R set up:

- more code needed, as fast ? (probably)
- refresh policies (V3) even more flexible
- output is directly available in R for further processing

Annual review, selective editing and manual fine tuning

Step 1: comparing old and new sets of parameters (classical "current"vs "automatic" reestimation)

- easy in both contexts using R (reading W with RJDemetra) :
- score computation with diagnostics : JDCruncher or adapted version to R objects

Step 2: select important series and fine-tune manually

- classical Workspace GUI for manual expertise: significant asset
- R setup: visual feedback not as rich, nor multi-layer, nor as easy to navigate as GUI

Contents

On production in R

Main asset of WS-GUI-Cruncher set up:

- GUI rich and multi-level feedback for manual fine tuning
- use data structure rigidity at your advantage (easier to keep track of production with workspaces ?)

Assets of "Full R set up"

- portability
- direct availability of objects in R

Upgrading from v2 to v3

What is new ?

- Tools : tests, arima estimation
- More specific and `fast()` functions
- Refresh policies, even more flexible

Upgrading code from v2 to v3

Cost of code conversion

- global functions very similar (arguments)
- customization process significantly different
- organisation of stored objects significantly different

Future developments

what is missing ?

- functions for handling workspaces in v3
- update/reframe auxiliary packages (rjdworkspace, ex ggdemetra3)

. . . open discussion

Possible Contributions

- Testing and reporting issues (test_rjd3 repo on GitHub)
- Developing new tools (other packages, new functions, etc.)

Resources and documentation

Webinar Resources on GitHub:

- slides with code (rmd files)
- additional references: Beamers, Working Paper on v2 set of tools (jan 2021) will be updated to v3 in 2023

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/

Restricted scope : Chapters on SA (incl HF) and Chapters on Tools (GUI, R packages and plug-ins) It will grow from there...

BLOG JDemetra+ universe

We are starting a Blog: **JDemetra+ universe**, the missing piece between Cros Portal and GitHub pages
https://jdemetra-universe-blog.netlify.app/

- can be used for problem/solution/insights sharing (comments available if logged into GitHub)
- guest posts welcome
- we will link "all" presentations about JDemetra+ in conferences / workshops * (If you give a talk about JD+ let us know. . .)

Thank you for your attention

Packages **R**:

- palatej/rjd3toolkit
- palatej/rjd3x13
- palatej/rjd3tramoseats
- palatej/rjdemetra3

- palatej/rjdfilters
- palatej/rjd3sts
- palatej/rjd3stl
- palatej/rjd3highfreq
- palatej/rjd3bench
- AQLT/ggdemetra3