Istat | Istituto Nazionale di Statistica

Frankfurt am Main, April 24, 2024

TSACE-TSAUG Meeting # 10

RJDemetra tools for statistical production

ALESSANDRO PIOVANI | alessandro.piovani@istat.it

Istat | Directorate For Methodology and Statistical Process Design

Presentation Overview

- **Pt.1: Tool for specification conversion**
  - JD_JSON format
  - TRAMO-SEATS (Gomez & Maravall) to RJDemetra (v2) / JDemetra+

- **Pt.2: RJDemetra Processor: an RJDemetra processing pipeline ready for you**
  - Architecture
  - Adaptable input interfaces to suit your context

Pt.1: Tool for specification conversion

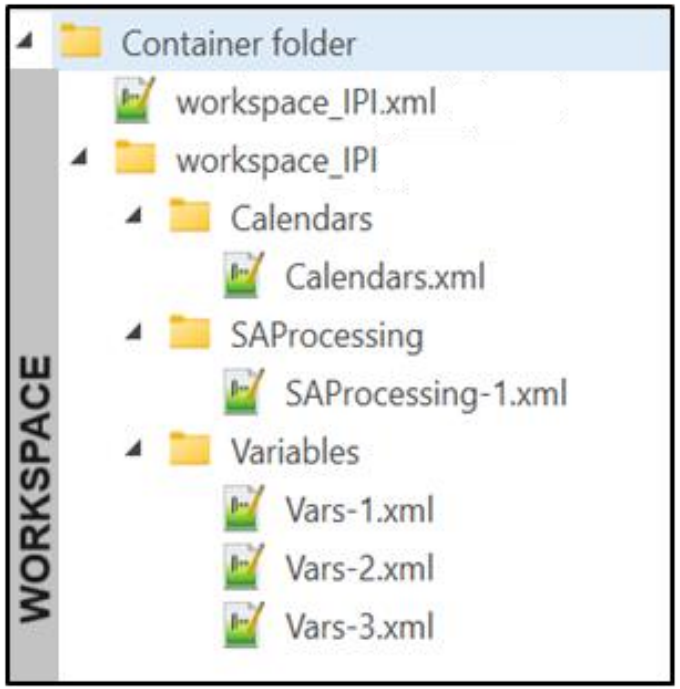
(TRAMO-SEATS to JDemetra+ and RJDemetra)

Istat | Istituto Nazionale di Statistica

Specifications in JDemetra+ and RJDemetra

JDemetra+ specifications can be retrieved from:

- the workspace (XML files into nested folders)
- .cfgx files
- .RData files (only RJDemetra and rjd3)



Istat

Why do we need this?

We found **difficulties** in handling specifications/workspaces **in memorization stage**:

- limited transparency
  - workspace XML are verbous
  - .cfgx or .RData files are not human-readable
- our database stores individual time series information, workspaces are designed for multiple time series
- one workspace for time-series is not practical for seasonal adjustment in JD+
- need a directory storage system (on filesystem or BLOB type field on DB) for workspaces
- workspaces store data, specifications, and external regressors together
- difficulties for the users in refreshing data (input time series and external regressors).
  - JD+ providers keep searching for data in the exact same location where they were originally loaded

JD_JSON format

- **JSON format**:
  - simple and widely recognized format → many libraryes in any languages to handle it
  - JavaScript types to represent the data → similar and easy to be mapped in R/Java types
  - "key":value pairs, divided by comma, enclosed in { }; arrays enclosed in [ ]; nested JSONs allowed
  - easy to read and write for users

- **Key-values specifications format:**
  - same as **c("SA_spec", "TRAMO_SEATS")** class attributes from **RJDemetra** (v2)
  - **little additions**
    - series name → to link specification and data
    - external regressors info → information to retrieve ext.reg., preferred to values for readability
  - comments with JavaScript/Java format (// inline, /* … */ multiline)

Istat

JD_JSON: examples

```
{
"series_name" : "FATEXP_10",
"spec"        : "RSA0",
"transform.function"      : "Log", // inline Comment
"usrdef.outliersEnabled" : true,
"usrdef.outliersType"     : ["AO",          "LS"],
"usrdef.outliersDate"     : ["2010-06-01", "2009-01-01"],
"arima.mu"                : false
}
```

```
{
"series_name" : "C_DEFL",
"spec"        : "RSA0",
"transform.function"      : "Log",
"usrdef.outliersEnabled" : true,
"usrdef.outliersType"     : ["AO",          "LS",          "AO",          "AO",          "AO",          "AO"],
"usrdef.outliersDate"     : ["2007-12-01", "2008-11-01", "2020-03-01", "2020-04-01", "2020-05-01", "2020-06-01"],
"userdef.varFromFile"     : true,

"userdef.varFromFile.infoList": [ { "file_name": "tdu02m.txt", "start":"2002-01-01", "frequency":12},
                                  { "file_name": "lym_02.txt", "start":"2002-01-01", "frequency":12}],
"usrdef.varEnabled"       : true,
"usrdef.varType"          : ["Calendar", "Calendar"],
"tradingdays.option"      : "UserDefined",
"arima.mu"                : false
}
```


Istat

JD_JSON: attributes borrowed from RJDemetra class c("SA_spec", "TRAMO_SEATS")

SOURCE: RJDemetra documentation

`tramoseats_spec` *TRAMO-SEATS model specification*

Description

Function to create (and/or modify) a `c("SA_spec", "TRAMO_SEATS")` class object with the SA model specification for the TRAMO-SEATS method. It can be done from a pre-defined 'JDemetra+' model specification (a `character`), a previous specification (`c("SA_spec", "TRAMO_SEATS")` object) or a seasonal adjustment model (`c("SA", "TRAMO_SEATS")` object).

Usage

```
tramoseats_spec(
  spec = c("RSAfull", "RSA0", "RSA1", "RSA2", "RSA3", "RSA4", "RSA5"),
  preliminary.check = NA,
  estimate.from = NA_character_,
  estimate.to = NA_character_,
  estimate.first = NA_integer_,
  estimate.last = NA_integer_,
  estimate.exclFirst = NA_integer_,
  estimate.exclLast = NA_integer_,
  estimate.tol = NA_integer_,
  estimate.eml = NA,
  estimate.urfinal = NA_integer_,
  transform.function = c(NA, "Auto", "None", "Log"),
  transform.fct = NA_integer_,
  usrdef.outliersEnabled = NA,
  usrdef.outliersType = NA,
  usrdef.outliersDate = NA,
  usrdef.outliersCoef = NA,
  ...
```

Arguments

`spec` a TRAMO-SEATS model specification. It can be the 'JDemetra+' name (`character`) of a predefined TRAMO-SEATS model specification (see *Details*), an object of class `c("SA_spec", "TRAMO_SEATS")` or an object of class `c("SA", "TRAMO_SEATS")`. The default is `"RSAfull"`.

`preliminary.check` a `logical` to check the quality of the input series and exclude highly problematic series e.g. the series with a number of identical observations and/or missing values above pre-specified threshold values.

The time span of the series, which is the (sub)period used to estimate the regarima model, is controlled by the following six variables: `estimate.from`, `estimate.to`, `estimate.first`, `estimate.last`, `estimate.exclFirst` and `estimate.exclLast`; where `estimate.from` and `estimate.to` have priority over the remaining span control variables, `estimate.last` and `estimate.first` have priority over `estimate.exclFirst` and `estimate.exclLast`, and `estimate.last` has priority over `estimate.first`. Default= "All".

`estimate.from` a character in format "YYYY-MM-DD" indicating the start of the time span (e.g. "1900-01-01"). It can be combined with the parameter `estimate.to`.

`estimate.to` a `character` in format "YYYY-MM-DD" indicating the end of the time span (e.g. "2020-12-31"). It can be combined with the parameter `estimate.from`.

`estimate.first` `numeric`, the number of periods considered at the beginning of the series.

`estimate.last` `numeric`, the number of periods considered at the end of the series.

`estimate.exclFirst` `numeric`, the number of periods excluded at the beginning of the series. It can be combined with the parameter `estimate.exclLast`.

...

JD_JSON: additional attributes
with respect to RJDemetra c("SA_spec", "TRAMO_SEATS")

- **series_name**: a character string indicating the name of the time series. Mandatory, because it allows the matching between specification and rawdata.

- **userdef.varFromFile**: logical indicating whether user-defined variable data will be read from the files specified by *userdef.varFromFile.infoList*, instead of *userdef.var* (raw data). If TRUE, the *userdef.var* field is ignored; if FALSE, *userdef.varFromFile.infoList* is ignored. Default = FALSE.

- **userdef.varFromFile.infoList**: A vector of JSON elements, each with the attributes:
  - container: character string representing the name of the file/entity containing the external regressor data;
  - start: a character string in the format "YYYY-MM-DD" indicating the starting time of the external regressor data;
  - n_var: number of variables contained in the file (in general =1 or =6 for 6TD setting)

JD_JSON: additional attributes
with respect to RJDemetra c("SA_spec", "TRAMO_SEATS")

- **frequency**: numeric. Frequency of the data: E.g. frequency=4 → quarterly data frequency=12 → monthly data.
- **method**: "TS" for TRAMO-SEATS, "X" for X13 (X13 not implemented yet).
- **intervention_variables**: a vector of JSON elements, each one with the attributes:
  - delta: Numeric;
  - delta_s: Numeric;
  - D1DS: boolean if D1DS mode (force delta to 1 and delta_s to 0);
  - sequences: JSON array: every element has the fields
    - start: a character string in the format "YYYY-MM-DD"
    - end: a character string in the format "YYYY-MM-DD"
- **ramps**: a vector of JSON elements, each with the attributes:
  - start: a character string in the format "YYYY-MM-DD" indicating the starting time of the ramp;
  - end: a character string in the format "YYYY-MM-DD" indicating the end time of the ramp;
  - fixed_coef: a numeric or 0 if not set)

EX. of RAMPs: [{"start":"2020-01-01","end":"2020-12-31", "fixed_coef":0},
{"start":"2008-01-01","end":"2009-01-01", "fixed_coef":1}]

EX. of IVs: [{"delta":1,"delta_s":1,"seq":[{"start":"2001-01-01","end": "2001-12-31", D1DS: false }]},
{"delta":0.75,"delta_s":0,"seq":[{"start":"2004-01-01","end":"2005-12-31", D1DS: false}]}]

JD_JSON: additional attributes
with respect to RJDemetra c("SA_spec", "TRAMO_SEATS")

- **easterCoef**: fixed coefficient for easter. 0 if not set.

Istat

JD_JSON: additional attributes
with respect to RJDemetra c("SA_spec", "TRAMO_SEATS")

```
{
"series_name" : "FATEXP_10",
"spec"                : "RSA0",
"transform.function"      : "Log", // inline Comment
"usrdef.outliersEnabled" : true,
"usrdef.outliersType"     : ["AO",          "LS"],
"usrdef.outliersDate"     : ["2010-06-01", "2009-01-01"],
"arima.mu"                : false
}
```

Series name

External regressors information

```
{
"series_name" : "C_DEFL",
"spec"                : "RSA0",
"transform.function"      : "Log",
"usrdef.outliersEnabled" : true,
"usrdef.outliersType"     : ["AO",          "LS",          "AO",          "AO",          "AO",          "AO"],
"usrdef.outliersDate"     : ["2007-12-01", "2008-11-01", "2020-03-01", "2020-04-01", "2020-05-01", "2020-06-01"],
"userdef.varFromFile"     : true,

"userdef.varFromFile.infoList": [ { "file_name": "tdu02m.txt", "start":"2002-01-01", "frequency":12},
                                  { "file_name": "lym_02.txt", "start":"2002-01-01", "frequency":12}],
"usrdef.varEnabled"       : true,
"usrdef.varType"          : ["Calendar", "Calendar"],
"tradingdays.option"      : "UserDefined",
"arima.mu"                : false
}
```


Istat

JD_JSON: features

- Already documented in large part → see tramoseats_spec function in RJDemetra doc.
- Easy to read and write for users → no nested objects, apart from *userdef.varFromFile.infoList*
- Easy to be stored into a DB → TEXT/VARCHAR or JSON* field DB types

  * some DBMS allow querying fields directly from JSON!
- Easy to be handled in R and Java → many libraries available (R: jsonlite/rjson, Java: jackson)
  → easy to be read by RJDemetra
- Facilitates synthesis → like specifying arguments to the tramoseats_spec function → default specification to assign values not provided by the user (e.g. "*RSA0*")
  → tools to produce the full version (see slide 13)

Istat

From full to synthetic version and vice versa


```
{
"series_name":"FATEXP_15",
"spec":"RSA0",
"usrdef.outliersEnabled":true,
"usrdef.outliersType":["AO",          "LS"],
"usrdef.outliersDate":["2009-09-01", "2009-03-01"],
"arima.mu":false,
"arima.p":1,
"arima.d":0,
"arima.q":0,
"arima.bq":0
},
```



```
{
"series_name":"FATEXP_15",
"spec":"RSA0",
"preliminary.check":true,
"estimate.from":"NA",
"estimate.to":"NA",
"estimate.first":"NA",
"estimate.last":"NA",
"estimate.exclFirst":0,
"estimate.exclLast":0,
"estimate.tol":1e-07,
"estimate.eml":true,
"estimate.urfinal":0.96,
"transform.function":"None",
"transform.fct":0.95,
"usrdef.outliersEnabled":true,
...
"arima.mu":false,
"arima.p":1,
"arima.d":0,
"arima.q":0,
"arima.bp":0,
"arima.bd":1,
"arima.bq":0,
"arima.coefEnabled":false,
"arima.coef":"NA",
"arima.coefType":"NA",
"fcst.horizon":-2,
"seats.predictionLength":-1,
"seats.approx":"Legacy",
"seats.trendBoundary":0.5,
"seats.seasdBoundary":0.8,
"seats.seasdBoundary1":0.8,
"seats.seasTol":2,
"seats.maBoundary":0.95,
"seats.method":"Burman"
},
```


JD_JSON.R

- **from_reduced_to_full_JD_JSON_file(…)**
- **from_full_to_reduced_JD_JSON_file(…)**

Istat

Workflow & Software

Now:

Work in progress:

JD_JSON.R

- JD_JSON_from_virtual_workspace(…)
- JD_JSON_to_virtual_workspace(…)
- JD_JSON_to_TSplus (…)*
- from_reduced_to_full_JD_JSON_obj (…)
- JD_JSON_from_materialized_workspace(…)
- JD_JSON_to_materialized_workspace(…)
- JD_JSON_from_TSplus(…)*
- from_full_to_reduced_JD_JSON_obj(…)

interoperability with JD+

* = work in progress

Istat

Pt.2: RJDemetra Processor:
an RJDemetra processing pipeline ready for you

Istat | Istituto Nazionale di Statistica

General overview

JD_JSON format is a component of a larger processing system that is:

- **R/RJDemetra based**
- **Object Oriented** → S4 Object System for R
- **Modular architecture**

This system is the response of our needs of:

- **a semi-automated production pipeline, interoperable with the JD+ suite**
  → workspaces as interoperability tool
- **use RJDemetra API with many time series without "hardcoding" specifications**
  → JD_JSON format
- **a tool expandable and adaptable to many situations**
  → adapting to various input formats/producing custom output
- **a platform mantainable also by statisticians without expertise in Java**
  → building R prototypes converted into Java by IT teams

Building blocks

Responsibility

JD_JSON_file_processor.R

- JD_JSON_file_processor (input_data_reader, ext_reg_data_reader, spec_file_name, output_workspace_dir, series_to_proc_names)

Processing and output of the results

JD_JSON.R

Interaction with workspace and input-output of specifications

c ("Extended_tramoseats_spec")

- Extends c("SA_spec", "TRAMOSEATS") from RJDemetra
- Attributes
- Constructor(…)

- to_JD_JSON(…)
- from_JD_JSON(…)
- to_named_list(…)
- from_named_list(…)
- to_sa_spec(…)
- from_sa_spec(…)
- to_tramoseats_spec_args(…)

Interaction with RJDemetra

DATA_READERS

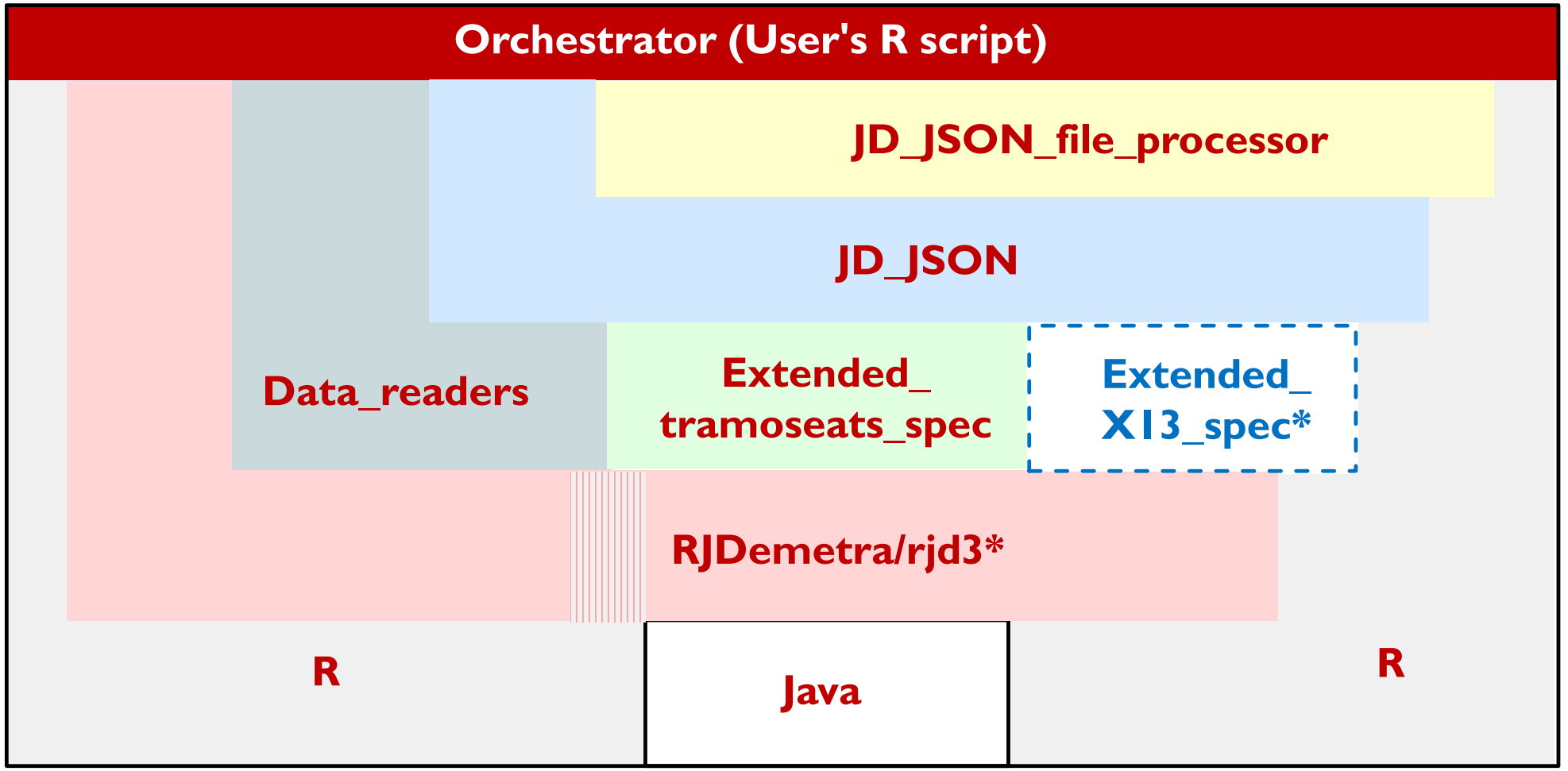
c ("Data_reader") **c ("Data_reader_ext_reg")**

Acquisition of the input data

Istat

Full JDProcessor Stack + possible extensions(*)

Notation: when a block lies on top of another, it means that it uses (or could use) it. → dependencies description



Adapt to your environment through PROVIDER interfaces

- To utilize JDProcessor in your environment, simply **implement the Provider and Provider_ext_reg interfaces** to read your data and external regressors.
  - flexible input, strict rules for the output
  - e.g. Provider_csv, Provider_txt, Provider_jdbc
- Then, pass the Providers to the processor.

Provider:

- Constructor's arguments:
  - input_source: "ANY"
- **read_data(…)** method:
  - **output: mts** (multivariate time series) obj., with time series names and dates
  - no specific input (…)

	FATEXP_10	FATEXP_11	FATEXP_13
2006-09-01	14,2	18,8	42,2
2006-10-01	14,9	21,3	43,6
2006-11-01	14	18,4	45,1
2006-12-01	13	15	45,8
2007-01-01	14,2	19,7	45,8
2007-02-01	14,9	20,2	44,4

PROVIDER_EXT_REG interface

Provider_ext_reg:

- Constructor: input_source (type : "ANY")
- **read_ext_reg_data** (var_info=NULL, time_series_info=NULL, …):
  - var_info = list with filename, start_date and frequency
  - time_series_info = series_name,
  - output: ts (time series) object

how to read data from var. info.

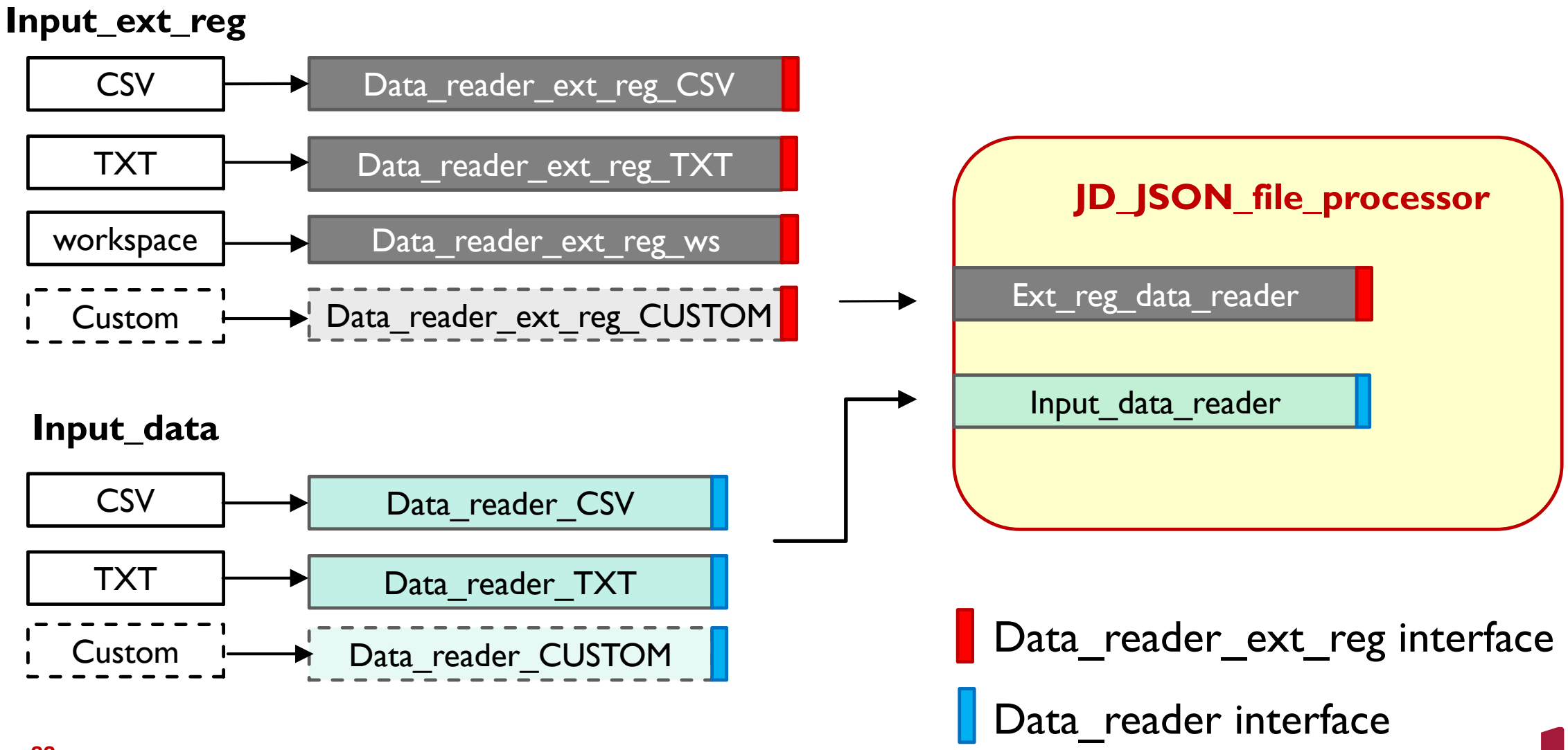
- **read_ext_reg_info** (var_info_container, …):
  - e.g. var_info_container = ext_reg_files_folder
  - **output: list of ext_var_info** for each ts →

how to produce var. info. (metadata)

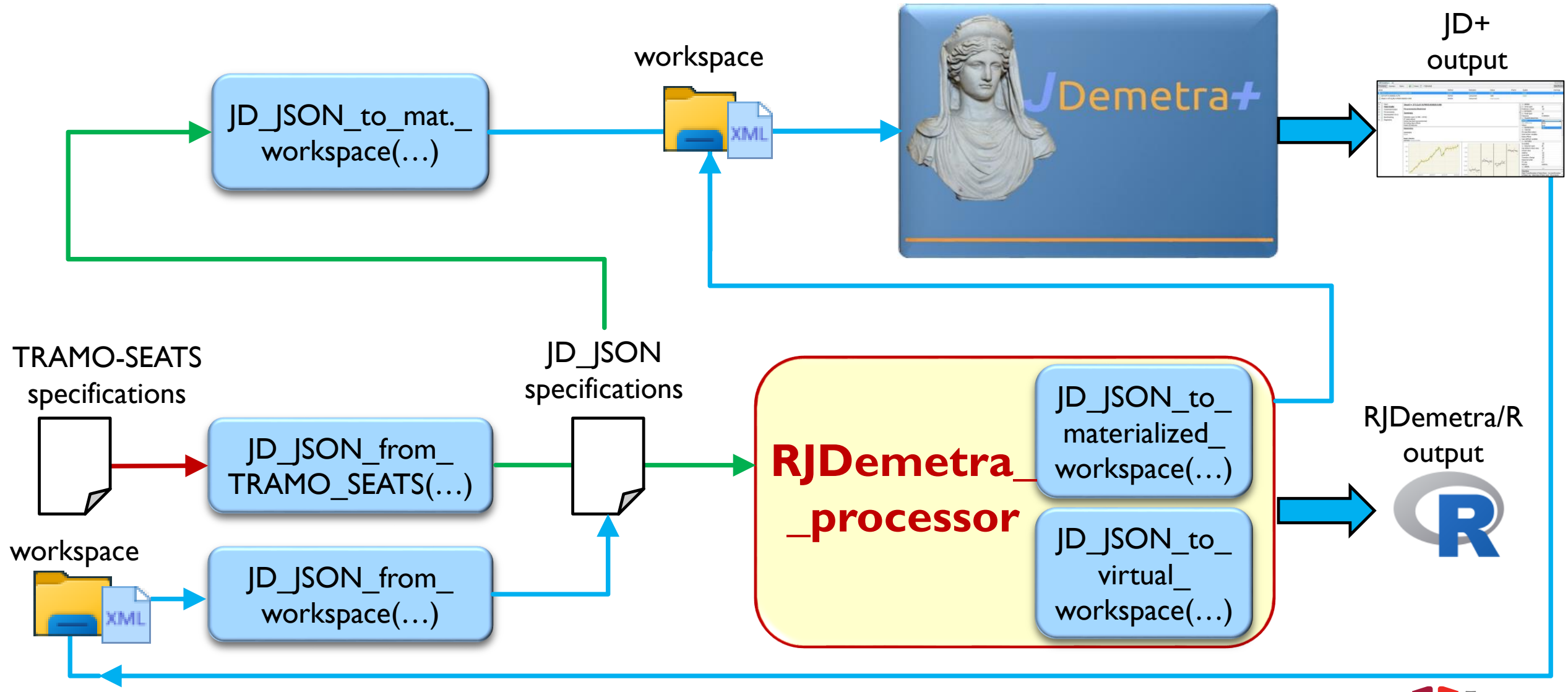
Name	Type	Value
all_jmodel_vars	list [105]	List of length 105
FATEXP_10	list [0]	List of length 0
C_DEFL	list [2]	List of length 2
[[1]]	list [3]	List of length 3
file_name	character [1]	'tdu02m.txt'
start	character [1]	'2002-01-01'
frequency	double [1]	12
[[2]]	list [3]	List of length 3
file_name	character [1]	'lym_02.txt'
start	character [1]	'2002-01-01'
frequency	double [1]	12

Istat

Adapt the processor with your Data Readers



Interoperability with JDemetra+ (GUI, Cruncher, …) through Workspace



Istat

Example of use: orchestrator.R

```
input_workspace_directory <- "C:\\Workspace-dir\\WS-FAT.xml" #Built with sa_ext plugin
input_data_file_name      <- "C:\\SITIC-FAT\\raw_data.csv"
regr_directory            <- "C:\\SITIC-FAT\\TS_regr" #Folder with external regressors
spec_file_name            <- "C:\\JD_JSON_specifications.txt"

diff <- TRUE  # Reduced JSON if diff=TRUE, Full JD_JSON format otherwise

############################## Operational flow ##############################

input_data_provider    <- Provider_csv_istat_format(input_data_file_name)
ext_reg_input_provider <- Provider_ext_reg_tsplus(regr_directory)

JD_JSON_from_materialized_workspace(input_workspace_directory, ext_reg_input_provider,
                                    JSON_file_name = spec_file_name, diff)

models <- JD_JSON_file_processor(input_data_provider, ext_reg_input_provider,
                                 spec_file_name, "output_workspace_container")#RJDemetra models

from_reduced_to_full_JD_JSON_file(spec_file_name, "JD_JSON_specifications_full.txt")
```


Istat

Future developments

- X13 integration
- Developing some default input providers
- Creating a rjd3-based version instead of RJDemetra
- Production of output formatting components
- Enhancing code performance by using RJDemetra functions that operate directly on Java objects rather than R ones
- Add other custom fields to JD_JSON ("Keep_ARIMA_coefficients_fixed", …) and relative functionalities in the processor
- Addition of detailed error messages
- Additional testing

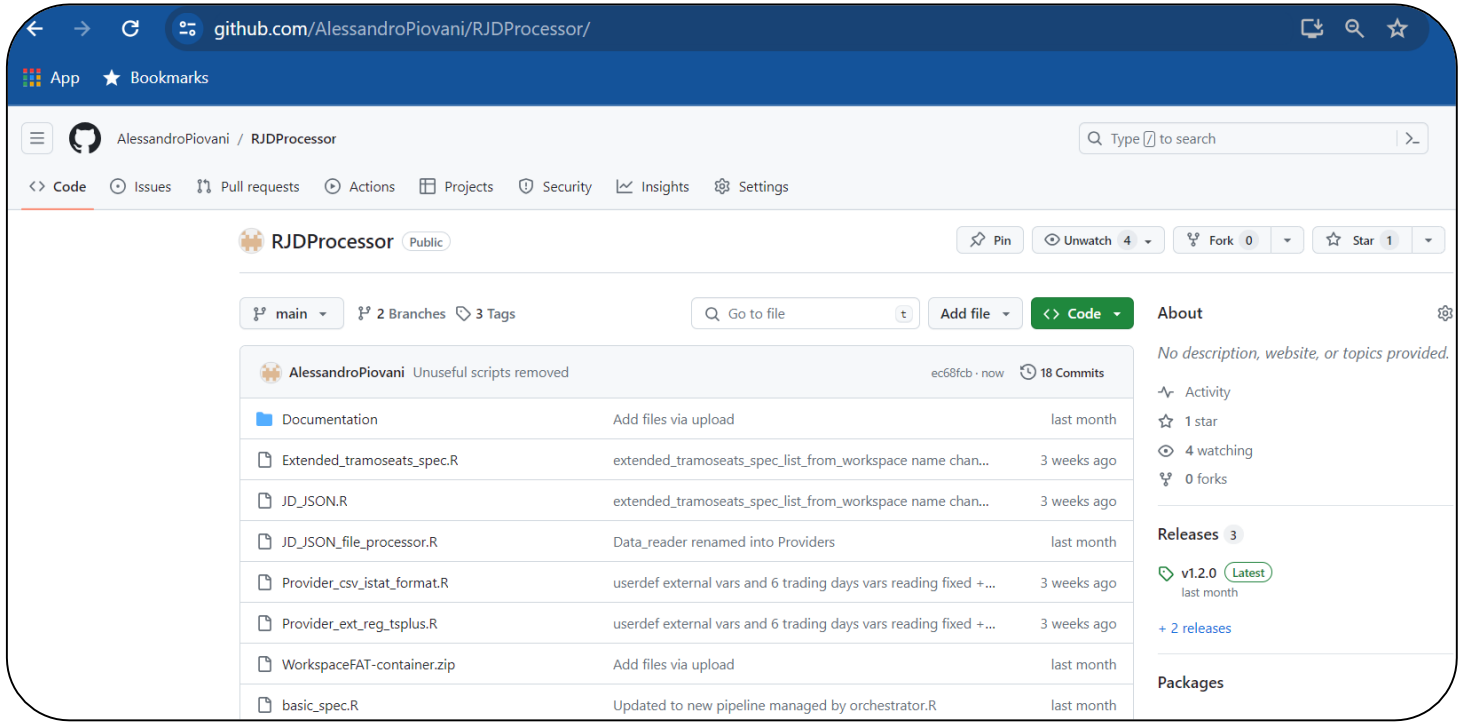
Source code

Source code is available on GitHub:

https://github.com/AlessandroPiovani/RJDProcessor

For information contact me at:

- alessandro.piovani@istat.it
- alessandro.piovani13@gmail.com



Thanks for your attention!

ALESSANDRO PIOVANI

alessandro.piovani@istat.it

Istat | Istituto Nazionale di Statistica