

# Package ‘RJDPprocessor’

January 29, 2025

**Type** Package

**Title** RJDPprocessor

**Version** 1.1.1

**Author** Alessandro Piovani

**Maintainer** Alessandro Piovani <alessandro.piovani@istat.it>,  
<alessandro.piovani13@gmail.com>

**Description** The rjdverse libraries are the officially recommended R software for seasonal adjustment in the European Central Bank and Statistical System. The RJDPprocessor library integrates the rjdverse packages into a fully R-based production pipeline, ready to be used and easily extendable by methodologists. It offers the capability to manage the entire seasonal adjustment process: acquisition, processing, storage, automation, and not just seasonal adjustment of the data. Processing of multiple time series is possible by storing their specifications in JSON files, and interoperability with other JDemetra+ software is guaranteed because RJDPprocessor can read workspaces and is able to produce them as an output.

RJDPprocessor also provides functions to manage workspaces, such as splitting a workspace containing multiple time series into individual single-series workspaces, which are suitable for storing in databases with single time series records. Functions to merge workspaces are also available.

**License** EUPL

**Encoding** UTF-8

**LazyData** true

**Imports** RJDemetra (>= 0.2.5),  
rjson (>= 0.2.21)

**Suggests** rjd3providers (>= 3.2.3),  
readxl (>= 1.4.3),  
roxygen2 (>= 7.2.3)

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Collate** import\_and\_interface\_definition.R  
Data\_reader\_csv.R  
Data\_reader\_csv\_istat\_format.R  
Data\_reader\_ext\_reg\_tsplus.R  
Data\_reader\_ext\_reg\_xlsx.R  
Data\_reader\_ext\_reg\_csv.R  
Data\_reader\_xlsx.R  
Data\_reader\_list.R  
Data\_reader\_xml.R

Data\_reader\_workspace.R  
 Extended\_tramoseats\_spec.R  
 JD\_JSON.R  
 JD\_JSON\_file\_processor.R  
 basic\_spec.R  
 utility\_functions.R  
 workspaces\_manager.R  
 report.R

## R topics documented:

check_data . . . . .	3
check_external_regressors . . . . .	3
compare_sa_ts . . . . .	4
compare_workspaces . . . . .	5
convert_numeric_matrix_to_mts . . . . .	6
create_diagnostic_report1 . . . . .	7
create_diagnostic_report2 . . . . .	8
Data_reader_csv . . . . .	8
Data_reader_csv_istat_format . . . . .	9
Data_reader_ext_reg_csv . . . . .	9
Data_reader_ext_reg_tsplus . . . . .	10
Data_reader_ext_reg_xlsx . . . . .	11
Data_reader_list . . . . .	11
Data_reader_workspace . . . . .	12
Data_reader_xlsx . . . . .	13
Data_reader_xml . . . . .	13
from_full_to_reduced_JD_JSON_file . . . . .	14
from_reduced_to_full_JD_JSON_file . . . . .	15
get_r_model_from_j_model . . . . .	15
get_single_ts_workspaces . . . . .	16
JD_JSON_file_processor . . . . .	17
JD_JSON_from_materialized_workspace . . . . .	18
JD_JSON_from_virtual_workspace . . . . .	20
JD_JSON_to_materialized_workspace . . . . .	21
JD_JSON_to_virtual_workspace . . . . .	22
merge_workspaces . . . . .	23
produce_fully_R_workspace . . . . .	24
read_data,Data_reader_csv-method . . . . .	25
read_data,Data_reader_csv_istat_format-method . . . . .	25
read_data,Data_reader_list-method . . . . .	26
read_data,Data_reader_workspace-method . . . . .	26
read_data,Data_reader_xlsx-method . . . . .	27
read_data,Data_reader_xml-method . . . . .	27
read_ext_reg_data,Data_reader_ext_reg_csv-method . . . . .	28
read_ext_reg_data,Data_reader_ext_reg_tsplus-method . . . . .	29
read_ext_reg_data,Data_reader_ext_reg_xlsx-method . . . . .	30
read_ext_reg_info,Data_reader_ext_reg_csv-method . . . . .	31
read_ext_reg_info,Data_reader_ext_reg_tsplus-method . . . . .	31
read_ext_reg_info,Data_reader_ext_reg_xlsx-method . . . . .	32
update_data . . . . .	33

**Index****34**


---

check_data	<i>Compare the data in a workspace with only one time series with given data</i>
------------	--

---

**Description**

This function compares the data in a workspace with only one time series with given raw data in array form

**Usage**

```
check_data(raw_data, ws_single_ts, raw_data_start = NA, raw_data_freq = NA)
```

**Arguments**

raw_data	Raw data to be compared with the ones in the workspace
ws_single_ts	The xml path of the workspace containing the data to be compared
raw_data_start	-optional- Default=NA, the starting date of the raw data in form "YYYY-MM-DD". If NA, the starting date is assumed to be the same as the workspace data
raw_data_freq	-optional- Default=NA, the frequency of the raw data ( e.g. 12=monthly, 4=quarterly). If NA, the frequency is assumed to be the same as the workspace data

**Value**

Boolean: TRUE if data are the same, FALSE otherwise

**Examples**

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
single_workspaces_path <- "splitted_workspaces_to_merge"
single_ws<-load_workspace(paste0(single_workspaces_path,"\\VATPIC\\VATPIC.xml"))
result<-check_data(c(1,2,3), single_ws)
setwd(original_directory)
```

---

check_external_regressors	<i>Verify the external regressors used in a workspace</i>
---------------------------	---

---

**Description**

This function check whether the external regressors used in a workspace are up to date (i.e. they cover from the beginning to the end of the time series and have the same frequency). The workspace must contain a single time series.

**Usage**

```
check_external_regressors(ws_single_ts)
```

**Arguments**

`ws_single_ts`     The xml path of the workspace containing the data to be checked

**Value**

Boolean: TRUE if ckeck is ok, FALSE if there are problems in the external regressors.

**Examples**

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
single_workspaces_path <- "splitted_workspaces_to_merge"
single_ws<-load_workspace(paste0(single_workspaces_path,"\\VATPIC\\VATPIC.xml"))
compute(single_ws)
result<- check_external_regressors(single_ws)
setwd(original_directory)
```

---

compare\_sa\_ts

*Compare the same models contained in two workspaces*

---

**Description**

This function compares the same models contained in two workspaces (old and new) plotting the respective sasonally adjusted series into a pdf file

**Usage**

```
compare_sa_ts(
  new_r_model = NA,
  new_model_workspace,
  old_model_workspace,
  materialized_ws_new = FALSE,
  materialized_ws_old = TRUE,
  java_processing_old_model = TRUE
)
```

**Arguments**

`new_r_model`     -optional- an R model obtained via `RJDemetra::get_model(workspace)`. If != NA, this model is used as new model in the comparison. Default = NA, `new_model_workspace` is used as new model

`new_model_workspace`     the workspace (relative/absolute) path of the .xml file or object used as new model in the comparison

`old_model_workspace`  
the workspace (relative/absolute) path of the .xml file or object used as old model in the comparison

`materialized_ws_new`  
-optional- Default=FALSE, boolean field stating whether the new workspace in the comparison is passed as a materialized or a virtual workspace (i.e. R object)

`materialized_ws_old`  
-optional- Default=TRUE, boolean field stating whether the new workspace in the comparison is passed as a materialized or a virtual workspace (i.e. R object)

`java_processing_old_model`  
-optional- Default=TRUE, use Java models for the internal processing of the old workspace (faster than R)

### Value

a "comparisons.pdf" file containig the plots of the seasonally adjusted time series of both the new and old workspaces. Series with the same series\_name are reported in the same plot

### Examples

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
input_workspace_materialized <- "workspace_test/workspace.xml"
virtual_workspace <- load_workspace(input_workspace_materialized)
compute(virtual_workspace)
compare_sa_ts(new_model_workspace = virtual_workspace,
              old_model_workspace = input_workspace_materialized ,
              materialized_ws_new=FALSE, materialized_ws_old=TRUE,
              java_processing_old_model=FALSE)
setwd(original_directory)
```

---

compare_workspaces	<i>Compare two RJDemetra workspaces</i>
--------------------	---

---

### Description

This function compares two RJDemetra workspaces, either provided as objects or file paths, and generates a report of the differences in the final seasonally adjusted series and likelihood BIC values.

### Usage

```
compare_workspaces(
  workspace1,
  workspace2,
  output_file = "differences_report.txt"
)
```

**Arguments**

workspace1	Either an RJDemetra workspace object or a file path to a workspace.
output_file	Character string specifying the path of the output report file. Default is "differences_report.txt".
patworkspace2	Either an RJDemetra workspace object or a file path to a workspace.

**Details**

The function loads the workspaces if they are provided as file paths, computes the models, and extracts the seasonally adjusted series and likelihood BIC. It then compares these values and records any differences in the output file.

Differences are checked for the last 36 observations of the seasonally adjusted series and the likelihood BIC values.

**Value**

The function does not return a value but writes a report to the specified file.

---

```
convert_numeric_matrix_to_mts
```

*Convert a numeric matrix to an mts object*

---

**Description**

This function takes a numeric matrix with dates as row names and series names as column names, and converts it to an mts (multivariate time series) object. It is good for converting data read by `data_readers` and `ext_reg_data_readers` into mts that could be useful as input as `usrdef.var` for RJDemetra functions `tramoseats()` and `X13()`

**Usage**

```
convert_numeric_matrix_to_mts(data_matrix, freq = NULL)
```

**Arguments**

data_matrix	A numeric matrix where row names are dates in "YYYY-MM-DD" format, and column names are the names of the time series.
freq	-optional- The frequency of the time series. If not provided, the function will try to infer the frequency based on the difference between the first two dates.

**Value**

An mts object with the given time series data.

**Examples**

```
input_data_file_name <- system.file("extdata", "CSV-FAS/grezzi_trim_FAS.csv", package = "RJProcessor")
input_data_reader    <- Data_reader_csv(input_source = input_data_file_name)
data_matrix <- input_data_reader@read_data()

# Convert the matrix to an mts object
mts_object <- convert_numeric_matrix_to_mts(data_matrix)
print(mts_object)
```

---

```
create_diagnostic_report1
```

*Create Diagnostic Report for Time Series Models*

---

**Description**

This function generates a diagnostic report for time series models, including BIC, p-values from Ljung-Box tests, and normality test results. It organizes the output into categories: all series, series with Ljung-Box problems, series with normality issues, and a summary of problematic series.

**Usage**

```
create_diagnostic_report1(workspace, output_file = "report.txt")
```

**Arguments**

workspace	A workspace object containing the time series models.
output_file	A string specifying the output file path where the report will be saved (default is "report.txt").

**Examples**

```
require(RJDemetra)

original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
ws_path <- "WorkspaceTUR-container/workspace-TUR.xml"

workspace <- load_workspace(file=ws_path)
create_diagnostic_report1(workspace, output_file = "report.out")
setwd(original_directory)
```

---

```
create_diagnostic_report2
```

*Create another Diagnostic Report for Time Series Models*

---

### Description

This function generates a diagnostic report for time series models. It shows for every time series a complete view, including regression coefficients with their T-statistics as well as LB, LB2, Normality tests and BIC

### Usage

```
create_diagnostic_report2(workspace, output_file = "series_info.txt")
```

### Arguments

workspace	A workspace object containing the time series models.
output_file	A string specifying the output file path where the report will be saved (default is "report.txt").

### Examples

```
require(RJDemetra)

original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJDProcessor")
setwd(extdata_directory)
ws_path <- "WorkspaceTUR-container/workspace-TUR.xml"

workspace <- load_workspace(file=ws_path)
create_diagnostic_report2(workspace, output_file = "report.out")
setwd(original_directory)
```

---

```
Data_reader_csv
```

*Constructor (R-like) of the Data\_reader object*

---

### Description

This function creates a Data\_reader object capable of reading data from CSV files and returning it using the read\_data() function.

### Usage

```
Data_reader_csv(input_source = NA, ...)
```

### Arguments

input_source	A string with file name (also with path).
--------------	---



**Value**

The Data\_reader\_csv object

**Examples**

```
input_data_file_name <- system.file("extdata", "CSV-FAS/grezzi_trim_FAS.csv", package = "RJDPProcessor")
input_data_reader <- Data_reader_csv(input_source = input_data_file_name)
input_data_reader@read_data()
```

---

Data\_reader\_csv\_istat\_format

*Constructor (R-like) of the Data\_reader object*

---

**Description**

This function creates a Data\_reader object capable of reading data from CSV files in ISTAT format and returning it using the read\_data() function. The ISTAT format is a csv file with dates in format YYYYqMM as rownames and time\_series names as colnames

**Usage**

```
Data_reader_csv_istat_format(input_source = NA, ...)
```

**Arguments**

input\_source     A string with file name (also with path).

**Value**

The Data\_reader\_csv\_istat\_format object

**Examples**

```
input_data_file_name <- system.file("extdata", "SITIC-TUR/grezziTUR.csv", package = "RJDPProcessor")
input_data_reader <- Data_reader_csv_istat_format(input_source = input_data_file_name)
#input_data_reader@read_data()
```

---

Data\_reader\_ext\_reg\_csv

*Constructor (R-like) of the Data\_reader object*

---

**Description**

This function creates a Data\_reader\_ext\_reg object capable of reading data from CSV external regressors files and returning it using the read\_ext\_reg\_data() function.

**Usage**

```
Data_reader_ext_reg_csv(input_source, ...)
```

**Arguments**

`input_source` A string with the input: e.g. a file name (also with path) if the input is a file.

**Value**

The `Data_reader_ext_reg_csv` object

**Examples**

```
require(RJDemetra)
input_workspace_xml <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDPProcessor")
input_data_file_name <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv", package = "RJDPProcessor")
regr_directory <- system.file("extdata", "CSV-TUR/regr", package = "RJDPProcessor")
ws <- load_workspace(file = input_workspace_xml)
compute(ws)
data_reader_ext_reg <- Data_reader_ext_reg_csv(regr_directory)
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
vars_matrix <- data_reader_ext_reg@read_ext_reg_data(all_model_ext_vars_info, "VATASC",
                                                    frequency=12)
```

---

`Data_reader_ext_reg_tsplus`

*Constructor (R-like) of the Data\_reader object*

---

**Description**

This function creates a `Data_reader_ext_reg` object capable of reading data from TRAMO-SEATS+ external regressors files and returning it using the `read_ext_reg_data()` function.

**Usage**

```
Data_reader_ext_reg_tsplus(input_source, ...)
```

**Arguments**

`input_source` A string with the input: e.g. a file name (also with path) if the input is a file.

**Value**

The `Data_reader_ext_reg_tsplus` object

**Examples**

```
require(RJDemetra)
input_workspace_xml <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDPProcessor")
input_data_file_name <- system.file("extdata", "SITIC-TUR/grezziTUR.csv", package = "RJDPProcessor")
regr_directory <- system.file("extdata", "SITIC-TUR/regr", package = "RJDPProcessor")
ws <- load_workspace(file = input_workspace_xml)
compute(ws)
data_reader_ext_reg <- Data_reader_ext_reg_tsplus(regr_directory)
```

```
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
vars_matrix             <- data_reader_ext_reg@read_ext_reg_data(all_model_ext_vars_info, "VATASC",
                                                                frequency=12)
```

---

Data\_reader\_ext\_reg\_xlsx

*Constructor (R-like) of the Data\_reader object*


---

## Description

This function creates a `Data_reader_ext_reg` object capable of reading data from XLSX external regressors files and returning it using the `read_ext_reg_data()` function.

## Usage

```
Data_reader_ext_reg_xlsx(input_source, ...)
```

## Arguments

`input_source` A string with the input: e.g. a file name (also with path) if the input is a file.

## Value

The `Data_reader_ext_reg_tsplus` object

## Examples

```
require(RJDemetra)
input_workspace_xlsx <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDProcessor")
input_data_file_name <- system.file("extdata", "SITIC-TUR/grezziTUR.csv", package = "RJDProcessor")
regr_directory       <- system.file("extdata", "SITIC-TUR/regr", package = "RJDProcessor")
ws                   <- load_workspace(file = input_workspace_xlsx)
compute(ws)
data_reader_ext_reg  <- Data_reader_ext_reg_tsplus(regr_directory)
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
vars_matrix          <- data_reader_ext_reg@read_ext_reg_data(all_model_ext_vars_info, "VATASC",
                                                                frequency=12)
```

---

Data\_reader\_list

*Constructor (R-like) of the Data\_reader object*


---

## Description

This function creates a `Data_reader` object capable of reading data from a list and returning it using the `read_data()` function.

## Usage

```
Data_reader_list(input_source = NA, ...)
```

**Arguments**

`input_source`     A string with file name (also with path).

**Value**

The `Data_reader_csv` object

**Examples**

```
FATEXP_10_list <- list("series_name"="FATEXP_10", "dates"=c("2005-01-01", "2005-02-01", "2005-03-01"),
                      "values"=c(12, 15, 11.1))
C_DEFL_list   <- list("series_name"="C_DEFL",   "dates"=c("2001-01-01", "2001-02-01", "2001-03-01"),
                      "values"=c(99, 100, 99.1))
# ...
input_data_list <- list(FATEXP_10_list, C_DEFL_list) #add more time series if you want (here are 2)
input_data_reader <- Data_reader_list(input_source = input_data_list)
input_data_reader@read_data()
```

---

`Data_reader_workspace`     *Constructor (R-like) of the Data\_reader object*

---

**Description**

This function creates a `Data_reader` object capable of reading data from workspaces and returning them using the `read_data()` function.

**Usage**

```
Data_reader_workspace(input_source = NA, ...)
```

**Arguments**

`input_source`     A string with workspace xml file name (also with path).

**Value**

The `Data_reader_workspace` object

**Examples**

```
input_data_file_name <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml", package = "RJDPProc")
input_data_reader    <- Data_reader_workspace(input_source = input_data_file_name)
mts <- input_data_reader@read_data()
```

---

Data_reader_xlsx	<i>Constructor (R-like) of the Data_reader object</i>
------------------	---

---

**Description**

This function creates a Data\_reader object capable of reading data from XLSX files and returning it using the read\_data() function.

**Usage**

```
Data_reader_xlsx(input_source = NA, ...)
```

**Arguments**

input\_source     A string with file name (also with path).

**Value**

The Data\_reader\_xlsx object

**Examples**

```
input_data_file_name <- system.file("extdata", "XLSX-TUR/grezzi_trim_TUR.xlsx", package = "RJProcessor")
input_data_reader <- Data_reader_xlsx(input_source = input_data_file_name)
input_data_reader@read_data()
```

---

Data_reader_xml	<i>Constructor (R-like) of the Data_reader object</i>
-----------------	---

---

**Description**

This function creates a Data\_reader object capable of reading data from XLSX files and returning it using the read\_data() function.

**Usage**

```
Data_reader_xml(input_source = NA, ...)
```

**Arguments**

input\_source     A string with file name (also with path).

**Value**

The Data\_reader\_xml object

**Examples**

```
input_data_file_name <- system.file("extdata", "Prod.xml", package = "RJProcessor")
# NOTE: absolute paths are better for this Data_reader
input_data_reader <- Data_reader_xml(input_source = input_data_file_name)
#input_data_reader@read_data() # for reading the data
```

---

```
from_full_to_reduced_JD_JSON_file
```

*Print a JSON file with only the fields that differ from the basic spec*

---

## Description

This function prints a JSON string that contains only the fields of the JD\_JSON object that differ from the ones of the basic specification (i.e. "RSA0", "RSA1", ...)

## Usage

```
from_full_to_reduced_JD_JSON_file(  
  JD_JSON_file,  
  output_file_name = NA,  
  indent = TRUE,  
  basic_spec = NA  
)
```

## Arguments

JD_JSON_file	The name of the file in which the JD_JSON will be saved
output_file_name	-optional- The name of the file (optionally with path) in which the JD_JSON will be saved. If NA, the name of the file will be paste0(JD_DJSON_file,"_reduced")
indent	-optional- Default TRUE. Print each field of the JSON in a different row
basic_spec	-optional- i.e. "RSA0", "RSA1", ... . Default=NA. If NA, use the basic spec specified in the input JSON

## Value

Void. A JSON file is saved on the filesystem

## Examples

```
require(RJDemetra)  
original_directory <- getwd()  
extdata_directory <- system.file("extdata", package = "RJProcessor")  
JSON_file_name <- system.file("extdata", "specifications_example3_full.txt",  
                             package = "RJProcessor")  
  
setwd(extdata_directory)  
from_full_to_reduced_JD_JSON_file(JSON_file_name, "specification_new_out2.txt")  
setwd(original_directory)
```

---

```
from_reduced_to_full_JD_JSON_file
```

*Print a JSON file with all the JD\_JSON fields explicit*

---

### Description

This function prints a JSON string that contains all the fields of the JD\_JSON object explicitly defined.

### Usage

```
from_reduced_to_full_JD_JSON_file(  
  JD_JSON_file,  
  output_file_name = NA,  
  indent = TRUE  
)
```

### Arguments

JD_JSON_file	The name of the file in which the JD_JSON will be saved
output_file_name	-optional- The name of the file (optionally with path) in which the JD_JSON will be saved. If NA, the name of the file will be paste0(JD_DSON_file,"_full")
indent	-optional- Default TRUE. Print each field of the JSON in a different row

### Value

Void. A JSON file is saved on the filesystem with all the JD\_JSON fields

### Examples

```
original_directory <- getwd()  
extdata_directory <- system.file("extdata", package = "RJDPProcessor")  
JSON_file_name <- system.file("extdata", "specifications_example2.txt", package = "RJDPProcessor")  
setwd(extdata_directory)  
from_reduced_to_full_JD_JSON_file(JSON_file_name)  
setwd(original_directory)
```

---

```
get_r_model_from_j_model
```

*Get an R list with model information from java model*

---

### Description

This function gets an R list with model information from java model

### Usage

```
get_r_model_from_j_model(j_model)
```

**Arguments**

j\_model                      a Java model obtained from RJDemetra

**Value**

a list containing all the models specifications

**Examples**

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJDPprocessor")
setwd(extdata_directory)
input_workspace <- "workspace_test/workspace.xml"
virtual_workspace <- load_workspace(input_workspace)
compute(virtual_workspace)
m <- get_jmodel(virtual_workspace)
r_model_list <- get_r_model_from_j_model(m)
setwd(original_directory)
```

---

```
get_single_ts_workspaces
```

*Get a single workspace for each series in a workspace*

---

**Description**

This function gets a single time series workspace for each time series SA model present in a given workspace

**Usage**

```
get_single_ts_workspaces(
  full_workspace,
  single_workspaces_path,
  compressed_ws = TRUE,
  clean_single_ws_directory = TRUE,
  from_TS_PLUS_plugin = TRUE
)
```

**Arguments**

full\_workspace    A workspace with multiple time series SA models

single\_workspaces\_path    The path in which the single workspaces will be stored

compressed\_ws    -optional- Default=TRUE, compress the single workspaces that will be created

clean\_single\_ws\_directory    -optional- Default=TRUE, delete the original workspace after creating the single series ones

from\_TS\_PLUS\_plugin    -optional- Default=TRUE, the original workspace is created by the SA\_ext\_plugin. In that case some internal adjustments are needed and they could make the computation slower. There are no problems if this field is TRUE but the ws is created without SA\_ext\_plugin



**Value**

A set of materialized workspaces in `single_workspaces_path`, one workspace for each time series  
SA model contained in `full_workspace`

**Examples**

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
single_workspaces_path <- "splitted_workspaces"
full_workspace <- "WorkspaceTUR-container\\workspace-TUR.xml"
compressed_ws <- FALSE
get_single_ts_workspaces(full_workspace, single_workspaces_path,
                          compressed_ws = compressed_ws )
setwd(original_directory)
```

---

JD\_JSON\_file\_processor

*Process a JD\_JSON file*

---

**Description**

This function processes a JSON file with JD\_JSON fields and returns a virtual workspace

**Usage**

```
JD_JSON_file_processor(
  input_data_reader,
  ext_reg_data_reader,
  spec_file_name,
  output_workspace_dir = NA,
  series_to_proc_names = NA,
  java_processing = TRUE
)
```

**Arguments**

`input_data_reader`  
A specific Data\_reader object (CSV, XLSX, ...) to read the input

`ext_reg_data_reader`  
A specific Data\_reader\_ext\_reg object (CSV, XLSX, ...) to read the external regressors

`spec_file_name` Name of the file (with path, if desired) containing the JD\_JSON to be processed (e.g. "specification\_new.txt")

`output_workspace_dir`  
-optional- Name of the directory that will contain the output workspace. Default=NA stores the workspace in a directory called "output\_workspace\_container"

`series_to_proc_names`  
-optional- vector of names of time series to be processed (e.g. c('VATASA','VATPIA'))  
. Default=NA: all the series are processed

java\_processing

-optional- Default=TRUE. Use only Java API (faster) for the internal computation

### Value

a virtual workspace, already processed

### Examples

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJDPProcessor")
setwd(extdata_directory)
spec_file_name <- "specifications_to_proc.txt"
input_workspace_directory <- "WorkspaceTUR-container/workspace-TUR.xml"
input_data_file_name <- "CSV-TUR/grezzi_trim_TUR.csv"
regr_directory <- "CSV-TUR/regr"
diff <- TRUE # Reduced JSON if diff=TRUE, Full JSON format otherwise

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

input_data_reader <- Data_reader_csv(input_source = input_data_file_name)
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)
JD_JSON_from_materialized_workspace(input_workspace_directory,
  ext_reg_input_data_reader, JSON_file_name = spec_file_name,
  diff=TRUE, java_processing=FALSE)
series_to_proc_names <- NA #c("FATEXP_13", "C_DEFL") # NA to process all
virtual_workspace <- JD_JSON_file_processor(input_data_reader = input_data_reader,
  ext_reg_data_reader = ext_reg_input_data_reader,
  spec_file_name = spec_file_name,
  output_workspace_dir = "output_workspace_container",
  series_to_proc_names = series_to_proc_names,
  java_processing = FALSE)
# set java_processor=TRUE to speed-up the operations, but it does not work with
# workspaces readed by sa-ext plugin
m <- get_model(virtual_workspace) #get directly the R model (slower)
from_reduced_to_full_JD_JSON_file(spec_file_name)
compare_sa_ts(new_model_workspace = virtual_workspace,
  old_model_workspace = input_workspace_directory ,
  materialized_ws_new=FALSE, materialized_ws_old=TRUE,
  java_processing_old_model=FALSE)
setwd(original_directory)
```

---

JD\_JSON\_from\_materialized\_workspace

*Turn model spec of a materialized workspace in JD\_JSON*

---

### Description

This function represent model specifications contained into a materialized workspace in JD\_JSON

**Usage**

```
JD_JSON_from_materialized_workspace(
  workspace,
  ext_reg_input_data_reader,
  regr_directory = NA,
  JSON_file_name = "JD_JSON_specification.txt",
  diff = TRUE,
  java_processing = FALSE
)
```

**Arguments**

<code>workspace</code>	Name of the workspace xml file (also with path).
<code>ext_reg_input_data_reader</code>	A <code>Data_reader_ext_reg</code> object, to read the external regressors in the desired format (csv, xlsx, tramoseats+, ...)
<code>regr_directory</code>	-optional- Name of the directory containing the sources (e.g. files) of the external regressors
<code>JSON_file_name</code>	-optional- Name of the JSON file to be created. If NA the file will be called "JD_JSON_specification.txt"
<code>diff</code>	-optional- if TRUE a reduced version of the JSON specification is produced; In the reduced version, fields with default values equals to the ones of the default specification (i.e. "RSA0", "RSA1", ...) are not reported Default=TRUE
<code>java_processing</code>	-optional- If TRUE, the function works internally with Java API (faster), otherwise it uses R API. Default=FALSE
<code>input_data_reader</code>	A <code>Data_Reader</code> object

**Value**

A JSON file saved on the filesystem

**Examples**

```
require(RJDemetra)

input_workspace <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                               package = "RJProcessor")
input_data_file_name <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv",
                                    package = "RJProcessor")
regr_directory <- system.file("extdata", "CSV-TUR/regr", package = "RJProcessor")
diff <- TRUE # Reduced JSON if diff=TRUE, Full JSON format otherwise

input_data_reader <- Data_reader_csv(input_source = input_data_file_name)
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)
JD_JSON_from_materialized_workspace(input_workspace, ext_reg_input_data_reader, JSON_file_name = "specificat
```

---

JD\_JSON\_from\_virtual\_workspace

*Turn model spec of a virtual (R) workspace in JD\_JSON*


---

## Description

This function represent model specifications contained into an R workspace in JD\_JSON

## Usage

```
JD_JSON_from_virtual_workspace(
  ws,
  ext_reg_input_data_reader,
  JSON_file_name = "JD_JSON_specification.txt",
  diff = TRUE,
  java_processing = TRUE
)
```

## Arguments

ws	workspace R object.
ext_reg_input_data_reader	A Data_reader_ext_reg object, to read the external regressors in the desired format (csv, xlsx, tramoseats+, ...)
JSON_file_name	-optional- Name of the JSON file to be created. If NA the file will be called "JD_JSON_specification.txt"
diff	-optional- if TRUE a reduced version of the JSON specification is produced; In the reduced version, fields with default values equals to the ones of the default specification (i.e. "RSA0", "RSA1", ...) are not reported Default=TRUE
java_processing	-optional- If TRUE, the function works internally with Java API (faster), otherwise it uses R API. Default=TRUE
input_data_reader	A Data_Reader object
regr_directory	-optional- Name of the directory containing the sources (e.g. files) of the external regressors

## Value

A JSON file saved on the filesystem

## Examples

```
require(RJDemetra)

input_workspace_directory <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                         package = "RJProcessor")
input_data_file_name      <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv",
                                         package = "RJProcessor")
regr_directory            <- system.file("extdata", "CSV-TUR/regr", package = "RJProcessor")
```

```

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

input_data_reader      <- Data_reader_csv(input_source = input_data_file_name)
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)

original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJDProcessor")
setwd(extdata_directory)

ws <- load_workspace(file = input_workspace_directory)
JD_JSON_from_virtual_workspace(ws, ext_reg_input_data_reader, JSON_file_name = "specifications_new_out.txt",
setwd(original_directory)

```

---

JD\_JSON\_to\_materialized\_workspace

*Turn a JD\_JSON in a materialized workspace*

---

## Description

This function obtain a JD\_JSON file from a workspace stored in the filesystem (in a directory). See test folder for examples

## Usage

```

JD_JSON_to_materialized_workspace(
  workspace_dir = NA,
  JSON_file,
  input_data_reader,
  ext_reg_data_reader = NA,
  series_to_proc_names = NA
)

```

## Arguments

**workspace\_dir** -optional- the directory of the input workspace. Default = NA stores the workspace in a directory called "output\_workspace\_container"

**JSON\_file** Name of the JSON file that will be produced (also with path).

**input\_data\_reader**  
A Data\_Reader object

**ext\_reg\_data\_reader**  
-optional- A Data\_reader\_ext\_reg object, to produce the metadata of the external regressors (i.e. create the names of the csv rather than a xlsx files o other containers for external regressors) Default = NA, do not consider external regressors (discouraged)

**series\_to\_proc\_names**  
-optional- an array containing the name of the series to be included in the workspace among the ones present in the JD\_JSON file e.g. c("VATASA", "VATAIA")

## Value

void in R environment, a workspace materialized in the filesystem

## Examples

```
require(RJDemetra)
input_JD_JSON      <- system.file("extdata", "specifications_example1.txt", package = "RJDPProcessor")
input_data_file_name <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv", package = "RJDPProcessor")
regr_directory      <- system.file("extdata", "CSV-TUR/regr", package = "RJDPProcessor")
input_data_reader    <- Data_reader_csv(input_source = input_data_file_name)
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)
JD_JSON_to_materialized_workspace(workspace_dir="ws_out" ,input_JD_JSON, input_data_reader, ext_reg_input_data_reader)
```

---

JD\_JSON\_to\_virtual\_workspace

*Turn a JD\_JSON in a virtual workspace*

---

## Description

This function obtain a virtual workspace from a JD\_JSON file. See test folder for examples

## Usage

```
JD_JSON_to_virtual_workspace(
  JSON_file,
  input_data_reader,
  ext_reg_data_reader = NA,
  series_to_proc_names = NA
)
```

## Arguments

**JSON\_file**            Name of the JSON file to turn in a workspace (also with path).

**input\_data\_reader**            A `Data_Reader` object

**ext\_reg\_data\_reader**            -optional- A `Data_reader_ext_reg` object, to read the external regressors in the desired format (csv, xlsx, tramoseats+, ...) Default = NA, do not consider external regressors (discouraged)

**series\_to\_proc\_names**            -optional- an array containing the name of the series to be included in the workspace among the ones present in the JD\_JSON file e.g. c("VATASA", "VATAIA")

## Value

A virtual workspace

## Examples

```
require(RJDemetra)
input_JD_JSON      <- system.file("extdata", "specifications_example1.txt", package = "RJDPProcessor")
input_data_file_name <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv", package = "RJDPProcessor")
regr_directory      <- system.file("extdata", "CSV-TUR/regr", package = "RJDPProcessor")
input_data_reader    <- Data_reader_csv(input_source = input_data_file_name)
```

```
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)
ws <- JD_JSON_to_virtual_workspace(input_JD_JSON, input_data_reader, ext_reg_input_data_reader, series_to_pr
```

---

merge_workspaces	<i>Merge many workspaces in one</i>
------------------	-------------------------------------

---

## Description

This function gets merges many workspaces contained in a given folder into one workspace

## Usage

```
merge_workspaces(
  source_workspaces_path,
  merged_ws_name = "merged_ws",
  merged_ws_dir = NA,
  compressed = TRUE,
  delete_originals = TRUE,
  silent = TRUE
)
```

## Arguments

source_workspaces_path	The path in which to find all the workspcaes that will be merged
merged_ws_name	-optional- Default="merged_ws". The name of the workspace that will be created
merged_ws_dir	-optional- Default=NA, the path in which the merged workspace will be stored. If NA it will be stored in the current directory
compressed	-optional- Default=TRUE, workspaces to be merged are compressed
delete_originals	-optional- Default=TRUE, delete original workspaces after merging them
silent	-optional- Default=TRUE, do not print status bar and messages during the operations

## Value

A virtual (R) and a marerialized workspace containing all the multiprocessings and time series SA models of the original workspaces

## Examples

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
single_workspaces_path <- "splitted_workspaces_to_merge"
full_workspace_path <- "merged_ws_test"
compressed_ws <- FALSE
ws_merged <- merge_workspaces(source_workspaces_path=single_workspaces_path,
```

```
merged_ws_dir = full_workspace_path, compressed = TRUE,
delete_originals = FALSE, silent=TRUE)

setwd(original_directory)
```

---

```
produce_fully_R_workspace
```

*Process a JD\_JSON file*

---

## Description

This function processes a JSON file with JD\_JSON fields and returns a virtual workspace

## Usage

```
produce_fully_R_workspace(
  ext_reg_input_data_reader = NA,
  original_ws_xml,
  new_r_ws_folder
)
```

## Arguments

```
ext_reg_input_data_reader
  A specific Data_reader_ext_reg object (CSV, XLSX, ...) to read the external
  regressors. Default=NA.

original_ws_xml
  xml file (also with path) of the workspace to be transformed into a fully R
  workspace

new_r_ws_folder
  Name of the directory that will contain the output, fully R, workspace.
```

## Value

a virtual workspace, already processed

## Examples

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
original_ws_xml <- "WorkspaceTUR-container/workspace-TUR.xml"
new_r_ws_folder <- "output_fully_R_ws"
regr_directory <- "SITIC-TUR/regr"
ext_reg_input_data_reader <- Data_reader_ext_reg_tsplus(regr_directory)
produce_fully_R_workspace(ext_reg_input_data_reader, original_ws_xml, new_r_ws_folder)

extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
original_ws_xml <- "WorkspaceFAS-standard-container//FAS.xml"
new_r_ws_folder <- "output_fully_R_ws"
```



```

regr_directory <- "CSV-FAS//regr"
ext_reg_input_data_reader <- Data_reader_ext_reg_csv(regr_directory)
produce_fully_R_workspace(ext_reg_input_data_reader, original_ws_xml, new_r_ws_folder)
setwd(original_directory)

```

---

read\_data,Data\_reader\_csv-method

*Get the data from a Data\_reader\_csv*


---

### Description

This function returns the data from the input\_source of the object.

### Usage

```

## S4 method for signature 'Data_reader_csv'
read_data(object, ...)

```

### Value

data in form of numeric matrix, with rownames = dates (in string format, YYYY-MM-DD) and colnames = time series names (string)

### Examples

```

input_data_file_name <- system.file("extdata", "CSV-FAS/grezzi_trim_FAS.csv", package = "RJDPProcessor")
input_data_reader <- Data_reader_csv(input_source = input_data_file_name)
input_data_reader@read_data()

```

---

read\_data,Data\_reader\_csv\_istat\_format-method

*Get the data from a Data\_reader\_csv\_istat\_format*


---

### Description

This function returns the data from the input\_source of the object.

### Usage

```

## S4 method for signature 'Data_reader_csv_istat_format'
read_data(object, ...)

```

### Value

data in form of numeric matrix, with rownames = dates (in string format, YYYYqMM) and colnames = time series names (string)

### Examples

```

input_data_file_name <- system.file("extdata", "SITIC-TUR/grezziTUR.csv", package = "RJDPProcessor")
input_data_reader <- Data_reader_csv_istat_format(input_source = input_data_file_name)
input_data_reader@read_data()

```

---

read\_data,Data\_reader\_list-method

*Get the data from a Data\_reader\_list*


---

### Description

This function returns the data from the input\_source of the object.

### Usage

```
## S4 method for signature 'Data_reader_list'
read_data(object, ...)
```

### Value

data in form of numeric matrix, with rownames = dates (in string format, YYYY-MM-DD) and colnames = time series names (string)

### Examples

```
FATEXP_10_list <- list("series_name"="FATEXP_10", "dates"=c("2005-01-01","2005-02-01","2005-03-01"),
                      "values"=c(12, 15, 11.1))
C_DEFL_list   <- list("series_name"="C_DEFL", "dates"=c("2001-01-01","2001-02-01","2001-03-01"),
                      "values"=c(99, 100, 99.1))
#...
input_data_list <- list(FATEXP_10_list, C_DEFL_list) #add more time series if you want (here are 2)
input_data_reader <- Data_reader_list(input_source = input_data_list)
input_data_reader@read_data()
```

---

read\_data,Data\_reader\_workspace-method

*Get the data from a Data\_reader\_workspace*


---

### Description

This function returns the data from the input\_source of the object.

### Usage

```
## S4 method for signature 'Data_reader_workspace'
read_data(object, ...)
```

### Value

data in form of numeric matrix, with rownames = dates (in string format, YYYY-MM-DD) and colnames = time series names (string)

### Examples

```
input_data_file_name <- system.file("extdata","WorkspaceTUR-container/workspace-TUR.xml", package = "RJDPProc")
input_data_reader    <- Data_reader_workspace(input_source = input_data_file_name)
mts <- input_data_reader@read_data()
```

---

`read_data,Data_reader_xlsx-method`*Get the data from a Data\_reader\_csv*

---

**Description**

This function returns the data from the `input_source` of the object.

**Usage**

```
## S4 method for signature 'Data_reader_xlsx'  
read_data(object, ...)
```

**Value**

data in form of numeric matrix, with `rownames` = dates (in string format, YYYY-MM-DD) and `colnames` = time series names (string)

**Examples**

```
input_data_file_name <- system.file("extdata", "XLSX-TUR/grezzi_trim_TUR.xlsx", package = "RJDPProcessor")  
input_data_reader    <- Data_reader_xlsx(input_source = input_data_file_name)  
input_data_reader@read_data()
```

---

`read_data,Data_reader_xml-method`*Get the data from a Data\_reader\_xml*

---

**Description**

This function returns the data from the `input_source` of the object.

**Usage**

```
## S4 method for signature 'Data_reader_xml'  
read_data(object, ...)
```

**Value**

data in form of numeric matrix, with `rownames` = dates (in string format, YYYY-MM-DD) and `colnames` = time series names (string)

**Examples**

```
input_data_file_name <- system.file("extdata", "Prod.xml", package = "RJDPProcessor")  
# NOTE: absolute paths are better for this Data_reader  
input_data_reader    <- Data_reader_xml(input_source = input_data_file_name)  
input_data_reader@read_data() # for reading the data
```



`read_ext_reg_data,Data_reader_ext_reg_tsplus-method`  
*Read external regressors data*

### Description

This function reads data from external regressors and returns it as a numeric matrix with variable names as colnames and YYYY-MM-DD dates as rownames

## Usage

```
## S4 method for signature 'Data_reader_ext_reg_tsplus'
read_ext_reg_data(
  object,
  var_info = NULL,
  time_series_info = NULL,
  frequency = NA_integer_,
  ...
)
```

## Arguments

var_info	A string with file name (also with path).
time_series_info	A string with time series name in workspace name (also with path).
frequency	i.e. 12 = monthly data, 4 = quarterly data

## Value

a numeric matrix with variable names as colnames and YYYY-MM-DD dates as rownames

## Examples

[illegible]



---

read\_ext\_reg\_info,Data\_reader\_ext\_reg\_csv-method

*Read information about external regressors from a workspace*


---

### Description

This function returns a list of information about external regressors used in the models contained in a workspaces

### Usage

```
## S4 method for signature 'Data_reader_ext_reg_csv'
read_ext_reg_info(object, var_info_container, adjust_path = TRUE, ...)
```

### Arguments

var\_info\_container  
workspace xml file path

### Value

list() of information about external regressors

### Examples

```
require(RJDemetra)
input_workspace_xml <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDProcessor")
input_data_file_name <- system.file("extdata", "CSV-TUR/grezzi_trim_TUR.csv", package = "RJDProcessor")
regr_directory <- system.file("extdata", "CSV-TUR/regr", package = "RJDProcessor")
ws <- load_workspace(file = input_workspace_xml)
compute(ws)
data_reader_ext_reg <- Data_reader_ext_reg_csv(regr_directory)
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
```

---

read\_ext\_reg\_info,Data\_reader\_ext\_reg\_tsplus-method

*Read information about external regressors from a workspace*


---

### Description

This function returns a list of information about external regressors used in the models contained in a workspaces

### Usage

```
## S4 method for signature 'Data_reader_ext_reg_tsplus'
read_ext_reg_info(object, var_info_container, adjust_path = TRUE, ...)
```

**Arguments**

var\_info\_container  
workspace xml file path

**Value**

list() of information about external regressors

**Examples**

```
require(RJDemetra)
input_workspace_xml <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDProcessor")
input_data_file_name <- system.file("extdata", "SITIC-TUR/grezziTUR.csv", package = "RJDProcessor")
regr_directory <- system.file("extdata", "SITIC-TUR/regr", package = "RJDProcessor")
ws <- load_workspace(file = input_workspace_xml)
compute(ws)
data_reader_ext_reg <- Data_reader_ext_reg_tsplus(regr_directory)
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
```

---

read\_ext\_reg\_info, Data\_reader\_ext\_reg\_xlsx-method

*Read information about external regressors from a workspace*

---

**Description**

This function returns a list of information about external regressors used in the models contained in a workspaces

**Usage**

```
## S4 method for signature 'Data_reader_ext_reg_xlsx'
read_ext_reg_info(object, var_info_container, adjust_path = TRUE, ...)
```

**Arguments**

var\_info\_container  
workspace xml file path

**Value**

list() of information about external regressors

**Examples**

```
require(RJDemetra)
input_workspace_xlsx <- system.file("extdata", "WorkspaceTUR-container/workspace-TUR.xml",
                                   package = "RJDProcessor")
input_data_file_name <- system.file("extdata", "XLSX-TUR/grezzi_trim_TUR.xlsx", package = "RJDProcessor")
regr_directory <- system.file("extdata", "XLSX-TUR/regr", package = "RJDProcessor")
ws <- load_workspace(file = input_workspace_xlsx)
compute(ws)
```



```
data_reader_ext_reg      <- Data_reader_ext_reg_xlsx(regr_directory)
all_model_ext_vars_info <- data_reader_ext_reg@read_ext_reg_info(ws)
```

---

update_data	<i>Update the data of a workspace</i>
-------------	---------------------------------------

---

## Description

This function update the data of a workspace's time series basing on the data read by a Data\_reader object already initialized. The time series read by the Data\_reader must have the same colnames as the time series names of the workspace to produce an update

## Usage

```
update_data(workspace_xml_path, data_reader)
```

## Arguments

workspace_xml_path	Path of the xml file of the workspace whose data have to be updated
data_reader	A Data_reader object already initialized with its input source

## Examples

```
require(RJDemetra)
original_directory <- getwd()
extdata_directory <- system.file("extdata", package = "RJProcessor")
setwd(extdata_directory)
ws_xml_path <- "TUR_ws_test_container/merged_ws.xml"
dr <- RJProcessor::Data_reader_csv(input_source = "rawdata_TUR.csv")
# num_mat<-dr@read_data() # to check if the data are available
update_data(ws_xml_path, dr)
setwd(original_directory)
```

# Index

check\_data, [3](#)  
check\_external\_regressors, [3](#)  
compare\_sa\_ts, [4](#)  
compare\_workspaces, [5](#)  
convert\_numeric\_matrix\_to\_mts, [6](#)  
create\_diagnostic\_report1, [7](#)  
create\_diagnostic\_report2, [8](#)  
  
Data\_reader\_csv, [8](#)  
Data\_reader\_csv\_istat\_format, [9](#)  
Data\_reader\_ext\_reg\_csv, [9](#)  
Data\_reader\_ext\_reg\_tsplus, [10](#)  
Data\_reader\_ext\_reg\_xlsx, [11](#)  
Data\_reader\_list, [11](#)  
Data\_reader\_workspace, [12](#)  
Data\_reader\_xlsx, [13](#)  
Data\_reader\_xml, [13](#)  
  
from\_full\_to\_reduced\_JD\_JSON\_file, [14](#)  
from\_reduced\_to\_full\_JD\_JSON\_file, [15](#)  
  
get\_r\_model\_from\_j\_model, [15](#)  
get\_single\_ts\_workspaces, [16](#)  
  
JD\_JSON\_file\_processor, [17](#)  
JD\_JSON\_from\_materialized\_workspace, [18](#)  
JD\_JSON\_from\_virtual\_workspace, [20](#)  
JD\_JSON\_to\_materialized\_workspace, [21](#)  
JD\_JSON\_to\_virtual\_workspace, [22](#)  
  
merge\_workspaces, [23](#)  
  
produce\_fully\_R\_workspace, [24](#)  
  
read\_data, Data\_reader\_csv-method, [25](#)  
read\_data, Data\_reader\_csv\_istat\_format-method, [25](#)  
read\_data, Data\_reader\_list-method, [26](#)  
read\_data, Data\_reader\_workspace-method, [26](#)  
read\_data, Data\_reader\_xlsx-method, [27](#)  
read\_data, Data\_reader\_xml-method, [27](#)  
read\_ext\_reg\_data, Data\_reader\_ext\_reg\_csv-method, [28](#)  
read\_ext\_reg\_data, Data\_reader\_ext\_reg\_tsplus-method, [29](#)  
read\_ext\_reg\_data, Data\_reader\_ext\_reg\_xlsx-method, [30](#)  
read\_ext\_reg\_info, Data\_reader\_ext\_reg\_csv-method, [31](#)  
read\_ext\_reg\_info, Data\_reader\_ext\_reg\_tsplus-method, [31](#)  
read\_ext\_reg\_info, Data\_reader\_ext\_reg\_xlsx-method, [32](#)  
  
update\_data, [33](#)