Le cruncher et les packages {rjwsacruncher} et {JDCruncheR}

Désaisonnalisation avec JDemetra+

Anna Smyk

Insee
Mesurer pour comprendre

JD+
Time Series Software for Official Statistics

Sommaire I

Section 1

Cruncher

Le cruncher (1/2)

Le Cruncher est un module exécutable additionnel, pas un package R. Il peut être lancé via R, SAS...

Objectif: mise à jour d'un workspace et export des résultats (séries et diagnostics), sans avoir à ouvrir l'interface graphique.

Très utile dans un processus de production

Quelques liens

- pour télécharger le cruncher https://github.com/jdemetra/jwsacruncher/releases
  Le cruncher est déjà installé sous AUS.
- l'aide associée au cruncher
  https://github.com/jdemetra/jwsacruncher/wiki

Le cruncher (2/2)

Pour mettre à jour un workspace il faut :

- le cruncher
- un fichier de paramètres (cf utilisateurs SAS)
- un workspace valide (chemin vers les données)

L'utilisation d'un package R évite de faire un fichier de paramètres :

- **{rjwsacruncher}** (sur le CRAN) : Mise à jour du workspace et export des outputs
- **{JDCruncheR}** (sur le CRAN) : Créer un bilan qualité pour le workspace à partir des outputs

Section 2

Lancement avec R

Installation et chargement des packages

Pour installer les packages :

```
install.packages("JDCruncheR", "rjwsacruncher")
```

Pour charger les packages :

```
library("JDCruncheR")
library("rjwsacruncher")
```

Cruncher avec {rjwsacruncher} (1/2)

Trois options à paramétrer:
- `default_matrix_item` (diagnostics à exporter)
- `default_tsmatrix_series` (séries temporelles à exporter)
- `cruncher_bin_directory` (chemin vers le cruncher).

Pour afficher les valeurs :

```
getOption("default_matrix_item")
getOption("default_tsmatrix_series")
getOption("cruncher_bin_directory")
```

Utiliser la fonction `options()` pour les modifier. Par exemple :

```
options(
    default_matrix_item =
        c(
            "likelihood.aic", "likelihood.aicc",
            "likelihood.bic", "likelihood.bicc"
        ),
    default_tsmatrix_series = c("sa", "sa_f"),
```

Cruncher avec {rjwsacruncher} (2/2)

Une fois les trois options précédentes validées le plus simple est d'utiliser la fonction `cruncher_and_param()` :

```
cruncher_and_param() # lancement avec paramètres par défaut

cruncher_and_param(
    workspace = "D:/Campagne_CVS/ipi.xml",
    # Pour ne pas renommer les noms des dossiers exportés :
    rename_multi_documents = FALSE,
    policy = "lastoutliers"
)
```

Pour voir l'aide associée à une fonction, utiliser `help()` ou `?` :

```
?cruncher_and_param
help(cruncher_and_param)
```

Section 3

Le bilan qualité avec {**JDCruncheR**}

Bilan qualité avec {**JDCruncheR**} (1/4)

Le package {**JDCruncheR**} permet de calculer un bilan qualité à partir des diagnostics exportés depuis JDemetra+ ou via le cruncher (fichier demetra_m.csv). Les trois principales fonctions sont :

- `extract_QR()` qui permet d'extraire le bilan qualité à partir du fichier csv contenant l'ensemble des diagnostics de JDemetra+ ;
- `compute_score()` pour calculer un score dans le bilan qualité ;
- `export_xlsx()` permet d'exporter le bilan qualité.

Bilan qualité avec {JDCruncheR} (2/4) : Exemple

```
# Sélectionner le fichier demetra_m.csv
# exporté à partir du cruncher
QR <- extract_QR()
QR

# Pour comprendre comment le score est calculé
?compute_score
QR <- compute_score(QR, n_contrib_score = 3)

QR

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

Si l'on a plusieurs workspaces (ou multi-documents) on peut "rassembler" les bilans qualité en les empilant avec la fonction `rbind()` ou en créant un objet `mQR` avec la fonction `mQR_matrix()`

Bilan qualité avec {JDCruncheR} (3/4) : Exemple

On peut aussi ne pas prendre en compte les valeurs manquantes et mettre des conditions sur les indicateurs :

```
# Poids de l'indicateur oos_mse réduit à 1 si les autres
# indicateurs valent "Bad" ou "Severe"
condition1 <- list(
    indicator = "oos_mse",
    conditions = c(
        "residuals_independency",
        "residuals_homoskedasticity",
        "residuals_normality"
    ),
    conditions_modalities = c("Bad", "Severe")
)
BQ <- compute_score(
    x = BQ,
    n_contrib_score = 5,
    conditional_indicator = list(condition1),
    na.rm = TRUE
```

Bilan qualité avec {JDCruncheR} (4/4) : Exemple

```
QR1 <- extract_QR()
QR2 <- extract_QR()

mQR <- mQR_matrix(Bilan_1 = QR1, Bilan_2 = QR2)

# On calcule le score pour tous les bilans
mQR <- compute_score(mQR, n_contrib_score = 3)
export_xlsx(mQR, export_dir = "U:/")
```