# RProtoBuf: sérialiser pour communiquer

Tanguy BARTHELEMY*

**Résumé**

Comment sérialiser de manière optimisée afin de communiquer entre différents langages ?

Lorsqu'on travaille sur un projet, il est souvent nécessaire de transmettre des informations à d'autres outils ou langages. Cela peut se faire via des fichiers, des API, ou d'autres passerelles mais ces méthodes ne sont pas toujours optimisées et adaptées. Les **Protocol Buffers** (Protobuf - Google) sont des structures de données sérialisées et optimisées pour cet objectif précis. Cette présentation mettra en avant les avantages de cette approche et montrera comment l'implémenter facilement en R et avec le package **{RProtoBuf}**. J'illustrerai ces concepts à travers des exemples concrets, notamment les packages **R** du rjdverse qui utilisent cette technologie pour interagir avec les routines Java de JDemetra+ et profiter à la fois des performances de calcul de Java et de la flexibilité de R.

**Mots-clefs (3 à 5)** : Sérialisation - Data - Package

## Développement

### Introduction

Le package **{RProtoBuf}** (Eddelbuettel, Stokely, and Ooms (2016)) repose sur la technologie de **Protocol Buffers** (Protobuf - Google) pour sérialiser les données. La sérialisation consiste à transformer les données en un format facilement transportable quel que soit les objets en entrée. À l'inverse, ce format est moins lisible par l'humain. Un des principaux atouts du format Protobuf est qu'il est agnostique du langage, c'est-à-dire qu'il peut être utilisé indépendamment du langage de programmation.

Le rôle de {RProtoBuf} est d'interfacer ce format de données en permettant son écriture et sa lecture. Il fournit ainsi des fonctions permettant la conversion des fichiers Protobuf en objets **R** et inversement.

L'utilisation des Protocol Buffers présente plusieurs avantages dans la sérialisation des données. Tout d'abord, comme précisé précédemment, cette sérialisation est *universelle* et ne dépend pas d'un langage de programmation en particulier ce qui facilite la communication entre différents environnements (voir la section sur l'application Java). Ensuite, cette sérialisation est extrêmement efficace et rapide et particulièrement adapté pour des objets de grande taille. Enfin, grâce à {RProtoBuf}, les données sont automatiquement traduites en classes adaptées selon le langage de programmation cible.

### Méthodes

La configuration des Protocol Buffers se fait au moyen de fichier `.proto`. Ces fichiers spécifient :

- le nom des objets transmis,
- leur contenu (chaînes de caractères, entiers, booléens...),
- et dans quel ordre les objets sont envoyés et reçus.

En R, {RProtoBuf} permet de lire ces fichiers `.proto` et génère automatiquement les fonctions qui permettront de convertir les buffers en objets S4 sous R ainsi que de créer les buffers à partir d'objets R.

*Insee, tanguy.barthelemy@insee.fr

### Application

JDemetra+ (Smyk et al. (2025)) est un logiciel open source d'analyse des séries temporelles (ajustement saisonnier, détection d'outliers, nowcasting...). Ce logiciel est recommandé par Eurostat et à tous les instituts du système statistique européen (ESS) depuis 2015. Le développement de JDemetra+ a été motivé par l'envie de proposer aux utilisateurs une interface et un support stable de production et d'études des séries temporelles. Une galerie de packages *rjdverse* (Palate et al. (2025)) basé sur la version 3 de JDemetra+ a été développé. Ces packages permettent de faire le lien entre R et les algorithmes de JDemetra+.

Ces algorithmes sont écrits en Java. C'est pourquoi les packages R relatifs à JDemetra+ ont été développés à partir de cette structure et de ces routines de base pour ajouter une surcouche de connexion de JDemetra+ à l'environnement R. Cette connexion est effectuée grâce au Protocol buffer (le protocole de sérialisation de Google). Le Protobuf permet de faire le lien entre les programmes Java et l'environnement R. Il est nécessaire de décrire la structure de classe des programmes Java et d'expliciter un moyen de convertir chaque élément en objet R. Les objets R seront nativement écrits en S4 mais pourront être réécrits en S3 pour être plus facilement manipulables.

Nous allons présenter une application de ces buffers à la connexion entre Java et R. Cet interfaçage permet une connexion bien plus rapide que par l'utilisation du package **{rJava}** notamment pour des gros objets ou pour des aller-retours répétés entre R et Java.

### Conclusion

{RProtoBuf} utilise les Protocol Buffers pour sérialiser efficacement les objets R. Comme présenté, cette approche peut aussi permettre de communiquer avec d'autres langage (ici Java).

Les principaux avantages de cette méthode sont :

- Réduction des temps de transmission des données,
- Prise en charge de gros objets,
- Facilité de développement, avec une conversion automatique en objets S4,
- Interopérabilité multi-langages, simplifiant les interactions entre R et d'autres environnements.

## Références


Eddelbuettel, Dirk, Murray Stokely, and Jeroen Ooms. 2016. "RProtoBuf: Efficient Cross-Language Data Serialization in r." *Journal of Statistical Software* 71 (2): 1–24. https://doi.org/10.18637/jss.v071.i02.

Palate, Jean, Tanguy Barthelemy, Alain Quartier-la-Tente, Philippe Charles, Anna Smyk, and Corentin Lemasson. 2025. "Organisation Rjdverse." GitHub. https://github.com/rjdverse.

Smyk, Anna, Tanguy Barthelemy, Alain Quartier-la-Tente, and Karsten Webel. 2025. *JDemetra+ Documentation*. jdemetra-new-documentation.netlify.app. https://jdemetra-new-documentation.netlify.app/.