

# extendR

frictionless bindings for R and Rust

---

Mossa Merhi Reimert

2024-07-19

Department of Veterinary and Animal Sciences,  
University of Copenhagen

PhD Fellow in Veterinary Epidemiology, M.Sc. in (mathematical) Statistics

Thesis is on Agent-based modelling of African Swine Fever between wild boars and domestic pigs

Supervisors: Matt Denwood, Maya Grussmann, Anette Boklund

## extendr: Frictionless bindings for R and Rust

**Mossa Merhi Reimert** <sup>1</sup>, **Josiah D. Parry** <sup>2</sup>, **Matt Denwood** <sup>1</sup>, **Maya Katrin Gussmann** <sup>1</sup>, **Claus O. Wilke** <sup>3</sup>, **Ilia Kosenkov** <sup>4</sup>, **Michael Milton** <sup>5</sup>, and **Amy Thomason** <sup>6</sup>

**1** Section for Animal Welfare and Disease Control, Department of Veterinary and Animal Sciences, University of Copenhagen, Denmark **2** Environmental Systems Research Institute (Esri), Redlands, CA, USA **3** Department of Integrative Biology, The University of Texas at Austin, Austin, TX, USA **4** Independent researcher, Finland **5** Walter and Eliza Hall Institute of Medical Research, Australia **6** Atomic Increment Ltd., United Kingdom

<https://joss.theoj.org/papers/10.21105/joss.06394>

# What is extendR?

extendR is a Rust extension for R.

- Official documentation for extending R ([R-exts](#)) supporting C/C++ / Fortran
- Community extensions: Rcpp, cpp11, rJava, reticulate (python), RJulia



# About R

- R is an interpreted language written in C.
- R is the successor of S
- R data format supports encoding of missing values, NA (like arrow)

# FFI challenges

- R's C-API is built around an opaque pointer type `SEXP`.
- R has a garbage collector
- Errors in R induce C `longjmps`

Also,

- Compatibility with CRAN requires MSRV 1.67.

# Overview

Package	CRAN compatible?	Published	Repository
<code>rextendr</code>	✓	CRAN	<a href="https://github.com/extendr/rextendr">github/extendr/rextendr</a>
<code>extendr-api</code>	✓	crates.io	<a href="https://github.com/extendr/extendr">github/extendr/extendr</a>
<code>extendr-macros</code>	✓	crates.io	
<code>extendr-engine</code>	!	crates.io	
<code>libR-sys</code>	✓	crates.io	<a href="https://github.com/extendr/libR-sys">github/extendr/libR-sys</a>

We encourage and appreciate all issues, discussions, and PRs sent to any of these repositories.

# Getting Started

R users prefer R for everything.

- In interactive session, we have `rextendr::rust_source()` and `rextendr::rust_function()` to execute R code *now*.

## Happy path

- Embed native code in R packages

```
usethis::create_package("newPkg")  
rextendr::use_rextendr()
```

To update R-wrappers use:  
`rextendr::document()`

```
newPkg  
├── DESCRIPTION  
├── NAMESPACE  
├── R  
│   └── extendr-wrappers.R  
├── newPkg.Rproj  
└── src  
    ├── Makevars  
    ├── Makevars.ucrt  
    ├── Makevars.win  
    ├── entrypoint.c  
    ├── newPkg-win.def  
    └── rust  
        ├── Cargo.toml  
        └── src  
            └── lib.rs
```



# That's it!

This is all you need to get started with R and Rust via extendR.

- [User Guide on extendr.github.io](https://extendr.github.io)
- [Walk-through of writing a binding package for rust crate heck.](#)
- We have a (friendly!) Discord! <https://discord.gg/7pbsz8rRvB>
- Josiah Parry [has a YouTube-channel that features extendR .](#)
- [YouTube: Build a geohash R package using Rust](#)

# Examples of `extendr-api` code

## Passing scalar values to Rust

```
use extendr_api::prelude::*;
```

```
#[extendr]
```

```
fn plus_one(x: f64) → f64 { x + 1.0 }
```

- `x` is copied to Rust
- `#[extendr]` exports the function to the R-package

# Examples of `extendr-api` code

## Returning strings to R

```
/// @export  
#[extendr]  
fn hello_world() → &'static str {  
    "Hello world!"  
}
```

- `/// @export` exports the function to other R-packages

# Examples of extendr-api code

## Modifying data in-place

```
#[extendr]
fn zero_middle_element(values: &mut [i32]) {
    let len = values.len();
    let middle = len / 2;
    values[middle] = 0;
}
```

- R's C-API natively supports i32, f64, and u8 only.

# R types and NA awareness

## Scalar

`Rbool`, `Rint`, `Rfloat`, and `Rstr` are all NA aware wrappers around `i32`, `f64` and an analogue to `&str`.

E.g. in-place mutation for doubles is `&mut [Rfloat]`.

## Vectors

`Logicals`, `Integers`, `Doubles`, and `Strings` are wrappers around R's `logical()`, `integer()`, `numeric()`, and `character()`.

# Examples of extendr-api code

## Passing Rust data to R

```
#[derive(Debug)]  
struct Person {  
    name: String,  
    age: u32,  
}
```

# Examples of extendr-api code

```
#[extendr]
impl Person {
  fn new() → Self {
    Self {
      name: "".to_string(),
      age: 0 }
  }
  fn name(&self) → &str {
    self.name.as_str()
  }
  fn set_name(&mut self, name:
&str) {
    self.name =
name.to_string();
  }
}
```

## On the R side

```
> person ← Person$new()
> person$set_name("Jeff")
> person
<pointer: 0x105c04530>
attr(,"class")
[1] "Person"
```

# Examples of extendr-api code

## Passing R owned Rust types

```
#[extendr]
impl Person {
    fn older<'a>(&'a self, other: &'a Self) → &'a Self {
        if self.age > other.age {
            self
        } else {
            other
        }
    }
}
```

Usage: Support for method-chaining in R



# extendr-api feature: serde

```
#[derive(Serialize, Deserialize)]
struct Person {
    name: String,
    age: Option<u32>,
}
```

In Cargo.toml

```
[dependencies]
extendr-api = { version = '*', features = ["serde"] }
serde       = { version = "*", features = ["derive"] }
```

# extendr-api feature: serde

```
let mut jeff = Person::new();  
jeff.set_name("Jeff");  
serializer::to_robj(&jeff).unwrap()
```

This translates to a `list()` in R:

```
$name  
[1] "Jeff"
```

```
$age  
NULL
```

# Roadmap

We are looking for contributors!

Call for Roadmap discussion in [extendr/#783](https://github.com/r-lib/extendr/issues/783).

*My agenda is*

- Support `{vctrs}` style R objects called records
- Add built-in support for arrow
- Provide low-level binding tools for advanced R-package authors
- Add enum as R factors conversion
- Only protect Rust allocated R data
- Serialize owned types to bytes in R

**Thanks for your attention!**

[extendr.github.io](https://extendr.github.io)