extendr

frictionless bindings for R and Rust

Mossa Merhi Reimert

2024-07-17

Department of Veterinary and Animal Sciences,
University of Copenhagen

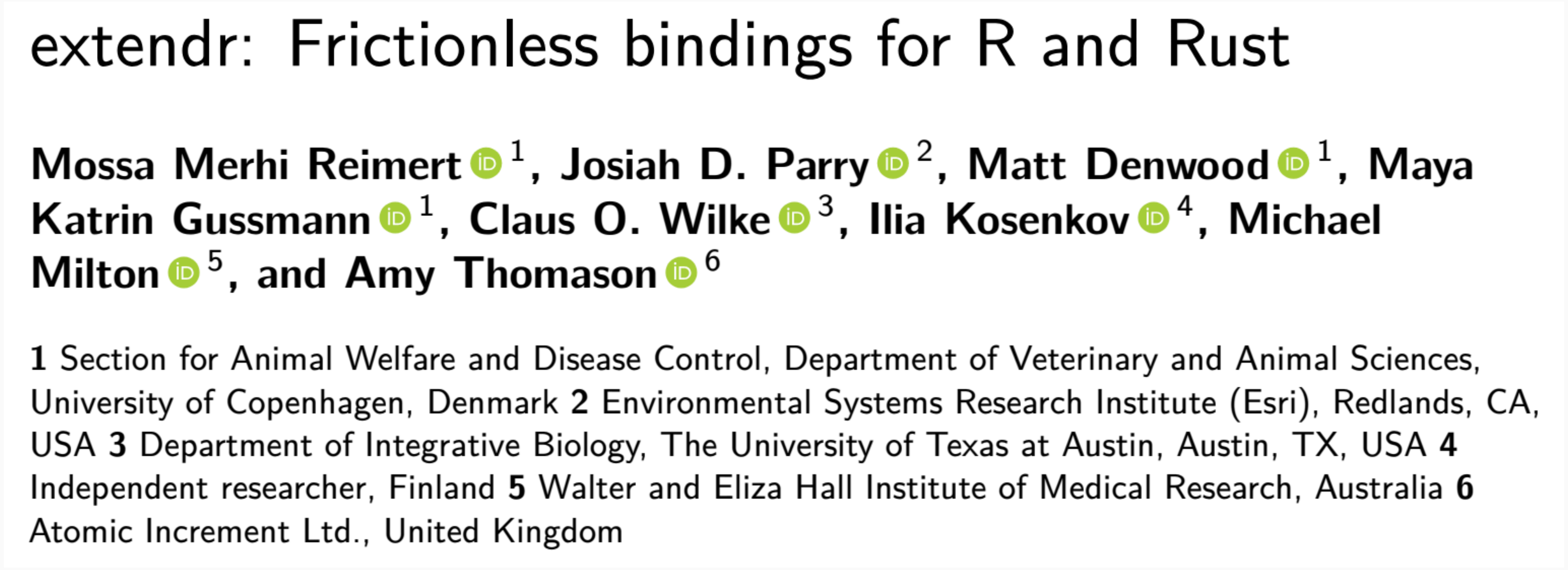
I’m Mossa

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 is published in JOSS



extendr: Frictionless bindings for R and Rust

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

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

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

R?

"R is a *free* software environment for *statistical* computing and graphics." – r-project.org.

- 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)

R is the language of many scientists, and CRAN is its main R-package repository.

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
`rextendr`	✓	CRAN	github/extendr/rextendr
`extendr-api`	✓	crates.io	github/extendr/extendr
`extendr-macros`	✓	crates.io	
`extendr-engine`	!	crates.io	
`libR-sys`	✓	crates.io	github/extendr/libR-sys

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

```r
usethis::create_package("newPkg")
rextendr::use_extendr()
```

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

```
.
├── DESCRIPTION
├── NAMESPACE
├── R
│   └── extendr-wrappers.R
├── hellorust.Rproj
└── src
    ├── Makevars
    ├── Makevars.ucrt
    ├── Makevars.win
    ├── entrypoint.c
    ├── hellorust-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
- Walk-through of writing a binding package for rust crate `heck`.
- We have a (friendly!) Discord! https://discord.gg/XAjKbDCW
- 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

```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

```rust
/// @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

```rust
#[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

```rust
#[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

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

Examples of `extendr-api` code

Passing R owned Rust types

```rust
#[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

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

In `Cargo.toml`

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

extendr-api feature: serde

```rust
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

Call for Roadmap discussion in extendr/#783.

My agenda is

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

Thanks for your attention

extendr.github.io