Exploring R and Rust in Bioinformatics

Online Developer Forum

Sun Wenjie, Mossa Merhi Reimert, Josiah Parry
Hosted by: Lluís Revilla
(Bioconductor Community Advisory Board)

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

28th July 2025

Thank you!

About Mossa

- Maintainer & Lead Developer of extendr
- PhD Veterinary Epidemiology
- Postdoc @ Københavns Universitet – University of Copenhagen

About Josiah

- Sr. Product Engineer @ Esri
- Spatial Statistician
- Building R packages for the R-ArcGIS Bridge

About Sun Wenjie

- Postdoc @ Institut Curie, Paris
- Computational Biologist & Bioinformatician
- Developer of tools in R, C++, and Rust
- Research focus: Cancer, stem cells, and data analysis

What is extendr?

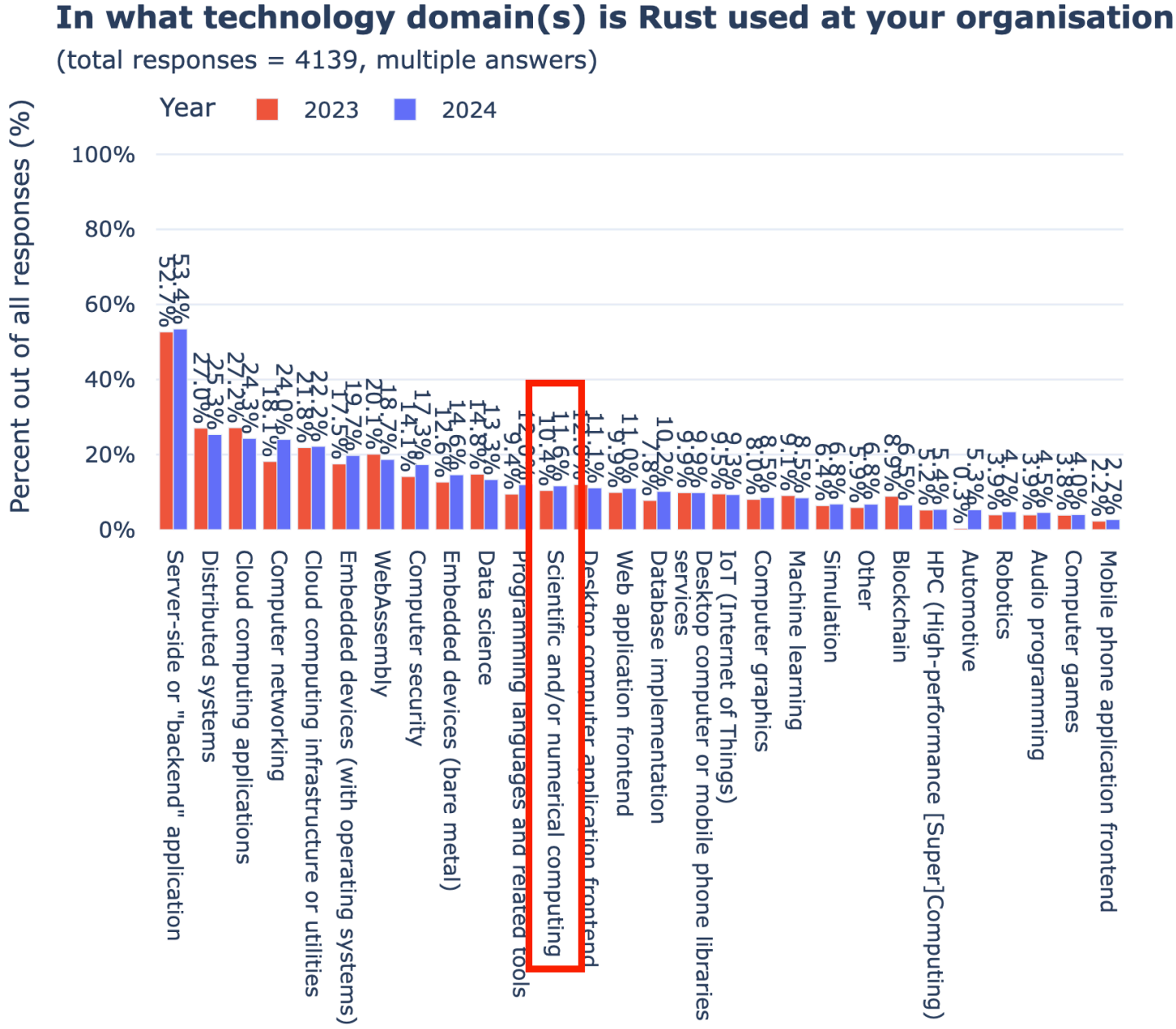
*extend*ing R with Rust

- Include Rust into R packages
- Akin to Rcpp & cpp11

What even is Rust?

- Low-level language
- Modern day equivalent to C++
- Memory safe-er
- Developer friendly
- No default garbage collector

Scientific computing with Rust is growing


In what technology domain(s) is Rust used at your organisation
(total responses = 4139, multiple answers)
Year
2023
2024
Percent out of all responses (%)
100%
80%
60%
40%
20%
0%
Server-side or "backend" application
Distributed systems
Cloud computing applications
Computer networking
Cloud computing infrastructure or utilities
Embedded devices (with operating systems)
WebAssembly
Computer security
Embedded devices (bare metal)
Data science
Programming languages and related tools
Scientific and/or numerical computing
Desktop computer application frontend
Web application frontend
Database implementation
Desktop computer or mobile phone libraries
IoT (Internet of Things)
Computer graphics
Machine learning
Simulation
HPC (High-performance [Super]Computing)
Blockchain
Other
Automotive
Robotics
Audio programming
Computer games
Mobile phone application frontend

extendr is comprised of:

- `{rextendr}` – usethis-like R package
- extendr-api – the "core" Rust library
- extendr-ffi – bindings to R's C API
- extendr-engine – R engine for embedding R into Rust

extendr packages

- 16 CRAN packages
- Many, many more on GitHub:

README MIT license

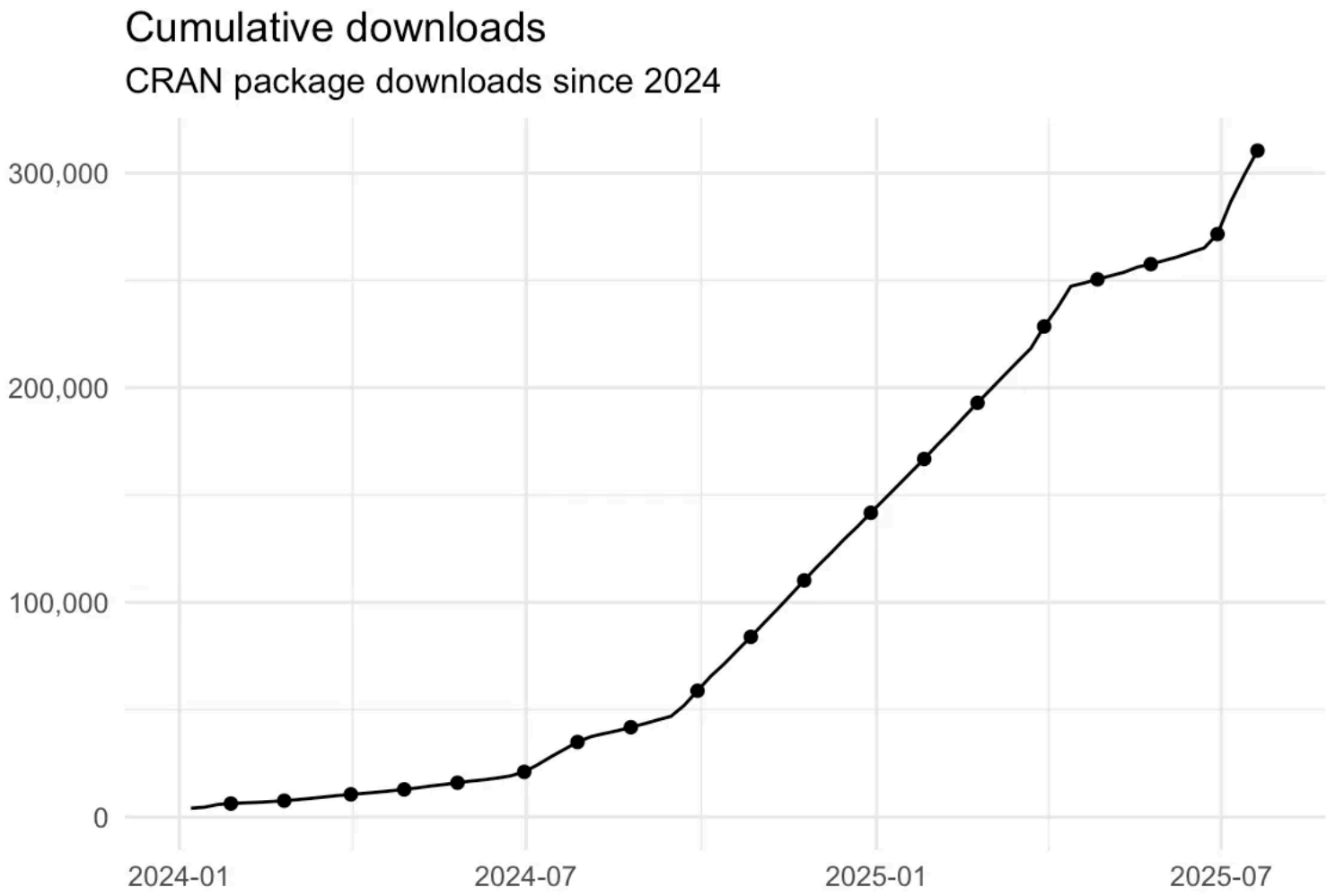
Awesome extendr

awesome | Last Updated 2025-07-27 | Total Downloads 381266

A curated list of R packages that use extendr and are currently published on CRAN.

- arcgis: ArcGIS Location Services Meta-Package
- arcgisgeocode: A Robust Interface to ArcGIS 'Geocoding Services'
- arcgisplaces: Search for POIs using ArcGIS 'Places Service'
- arcgisutils: ArcGIS Utility Functions
- arcpbf: Process ArcGIS Protocol Buffer FeatureCollections
- astgrepr: Parse and Manipulate R Code
- awdb: Query the USDA NWCC Air and Water Database REST API
- b64: Fast and Vectorized Base 64 Engine
- ciflyr: Reachability-Based Primitives for Graphical Causal Inference
- fcl: A Financial Calculator
- fio: Friendly Input-Output Analysis
- heck: Highly Performant String Case Converter
- orbweaver: Fast and Efficient Graph Data Structures
- rbm25: A Light Wrapper Around the 'BM25' 'Rust' Crate for Okapi BM25 Text Search
- rshift: Paleoecology Functions for Regime Shift Analysis
- rtiktoken: A Byte-Pair-Encoding (BPE) Tokenizer for OpenAI's Large Language Models
- SQLFormatteR: Format SQL Queries
- tergo: Style Your Code Fast
- tomledit: Parse, Read, and Edit 'TOML'
- unsum: Reconstruct Raw Data from Summary Statistics
- xactonomial: Inference for Functions of Multinomial Parameters
- ymd: Parse 'YMD' Format Number or String to Date
- zoomerjoin: Superlatively Fast Fuzzy Joins

380k downloads and counting



Why Rust?

Blazingly fast

Rust Is Blazingly Fast
(And We Won't Shut Up About It)

— The entire Rust community.

The compiler is nice

Source: https://programmerhumor.io/backend-memes/rust-3/

The compiler is nice


```
error[E0384]: cannot assign twice to immutable variable `i`
 --> src/main.rs:7:3
  |
4 |     let i = 0;
  |         -
  |         |
  |         first assignment to `i`
  |         help: make this binding mutable: `mut i`
...
7 |         i += 1;
  |         ^^^^^^ cannot assign twice to immutable variable

error: aborting due to previous error; 1 warning emitted

For more information about this error, try `rustc --explain E0384`.
```

Why You Should Stop What You're Doing Right Now and Learn Rust

Easy (enough) to refactor

r/rust • 2 yr. ago
Alchnator

so is Rust easy to refactor or bad to refactor?

Closed • 858 total votes

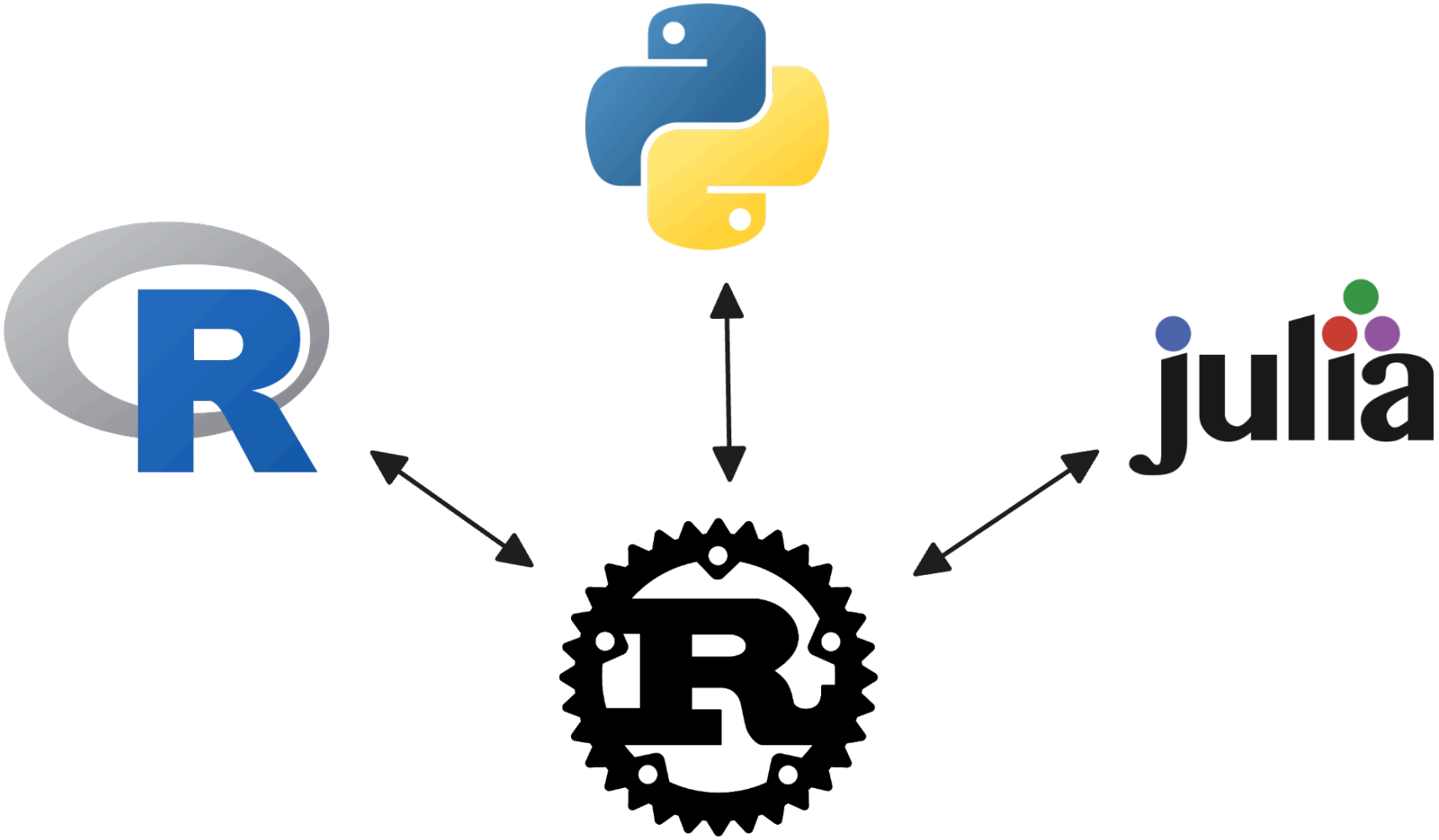
643	Rafactoring in Rust is Easy
215	Refactoring in Rust is Hard

Voting closed 2 years ago

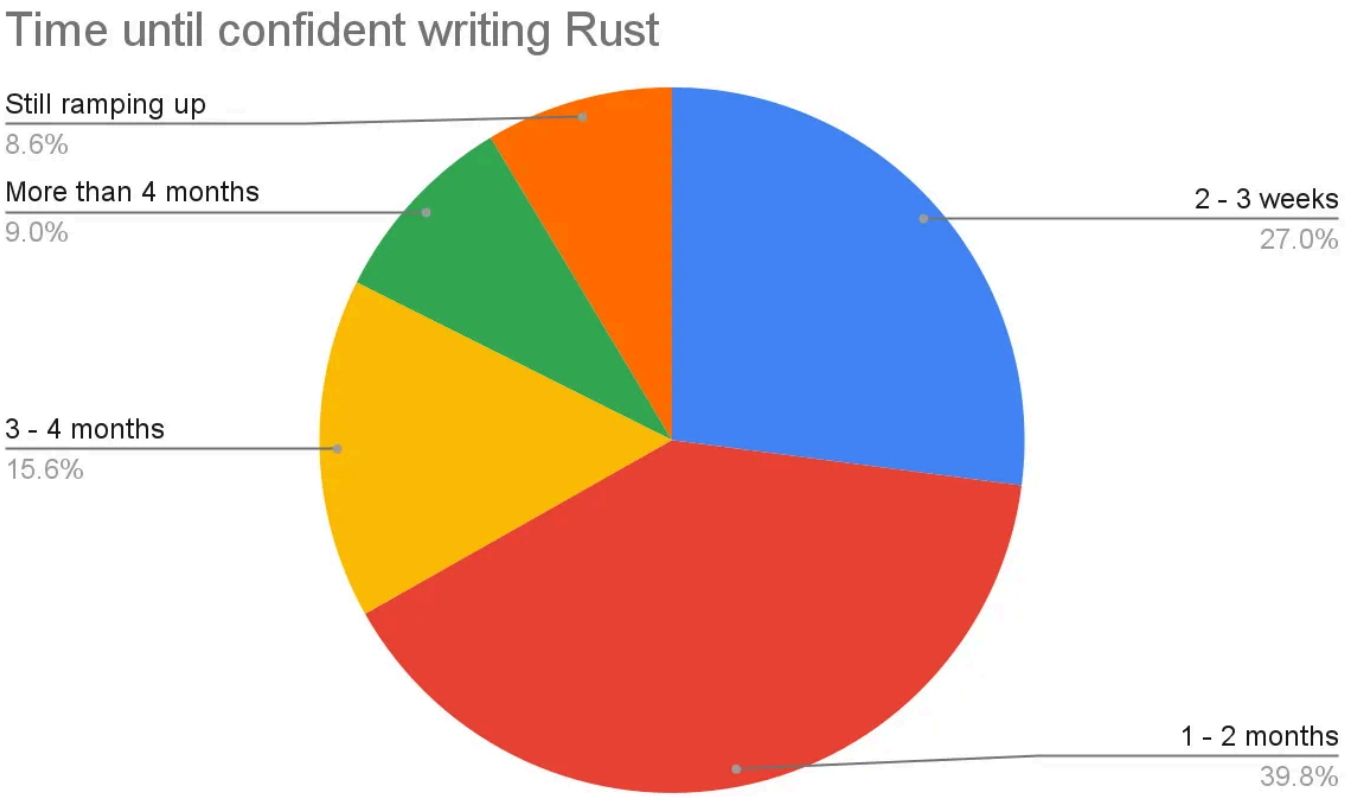
i'm curious to hear of people experiences with refactoring rust code, because I'm hearing total opposite opinions from different people.

Source: reddit poll

Why Rust?



Quick to pickup



A little skill goes a long way

Example: single

rust

```rust
#[extendr]
fn gh_encode(x: f64, y: f64, length: usize) → String {
    let coord = Coord { x, y };
    encode(coord, length).expect("Failed to encode the geohash")
}
```

Example: single

rust

```rust
#[extendr]
fn gh_encode(x: f64, y: f64, length: usize) -> String {
    let coord = Coord { x, y };
    encode(coord, length).expect("Failed to encode the geohash")
}
```

Example: single

rust

```rust
#[extendr]
fn gh_encode(x: f64, y: f64, length: usize) -> String {
    let coord = Coord { x, y };
    encode(coord, length).expect("Failed to encode the geohash")
}
```

Example: single

rust

```rust
#[extendr]
fn gh_encode(x: f64, y: f64, length: usize) → String {
    let coord = Coord { x, y };
    encode(coord, length).expect("Failed to encode the geohash")
}
```

Example: vectorize

rust

```rust
#[extendr]
fn gh_encode(x: &[f64], y: &[f64], length: usize) -> Vec<String>
{
  x
    .into_iter()
    .zip(y.into_iter())
    .map(|(xi, yi)| {
        let coord = Coord { x: xi, y: yi };
        encode(coord, length)
            .expect("Failed to encode the geohash")
    })
    .collect::<Vec<_>>()
}
```

Example: vectorize

rust

```rust
#[extendr]
fn gh_encode(x: &[f64], y: &[f64], length: usize) -> Vec<String>
{
  x
    .into_iter()
    .zip(y.into_iter())
    .map(|(xi, yi)| {
        let coord = Coord { x: xi, y: yi };
        encode(coord, length)
            .expect("Failed to encode the geohash")
    })
    .collect::<Vec<_>>()
}
```

Example: vectorize

rust

```rust
#[extendr]
fn gh_encode(x: &[f64], y: &[f64], length: usize) → Vec<String>
{
  x
    .into_iter()
    .zip(y.into_iter())
    .map(|(xi, yi)| {
        let coord = Coord { x: xi, y: yi };
        encode(coord, length)
            .expect("Failed to encode the geohash")
    })
    .collect::<Vec<_>>()
}
```

Example: parallelize

rust

```rust
#[extendr]
fn gh_encode(x: &[f64], y: &[f64], length: usize) → Vec<String>
{
  x
    .into_iter()
    .zip(y.into_iter())
    .par_bridge() // convert into a parallel iterator
    .with_min_len(1024) // set minimum parallel chunk length
    .map(|(xi, yi)| {
        let coord = Coord { x: xi, y: yi };
        encode(coord, length)
            .expect("Failed to encode the geohash")
    })
    .collect::<Vec<_>>()
}
```

extendr in bioinformatics

WebGestaltR

WebGestaltR 1.0.0

WebGestaltR

R-CMD-check failing

> [!IMPORTANT] The new version of WebGestaltR requires Rust, which must be installed on your device prior to installing or updating the package from CRAN. See the installation section for more information.

WebGestalt R package is the R version of our well-known web application tool WebGestalt (www.webgestalt.org) that has on average 27,000 users from 140 countries and territories per year and has also been cited 371 in 2016. The advantage of this R package is that it can be easily integrated to other pipelines or simultaneously analyze multiple gene lists.

github/bzhanglab/WebGestaltR

README GPL-3.0 license

R-CMD-check passing | Install-check passing | codecov 62%
GitHub v1.5.0 | CRAN v1.2.6 | downloads 1716

metabodecon

A framework for deconvolution, alignment and postprocessing of 1D NMR spectra, resulting in a data matrix of aligned signal integrals. The deconvolution part uses the algorithm described in Koh et al. (2009). The alignment part is based on functions from the 'speaq' package, described in Beirnaert et al. (2018) and Vu et al. (2011). A detailed description and evaluation of an early version of the package, 'MetaboDecon1D v0.2.2', can be found in Haeckl et al. (2021).

github/spang-lab/metabodecon

README GPL-3.0 license

R-CMD-check passing | Test-Install passing | r-universe 0.0.2

mdrb

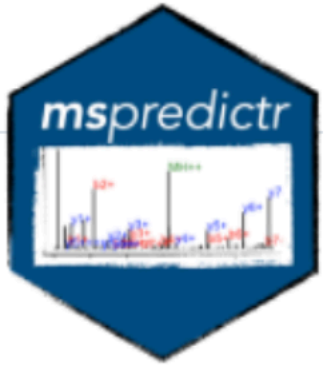
Provides a high-performance Rust backend for the metabodecon package, enabling efficient deconvolution, alignment, and post-processing of 1D NMR spectra. The package wraps optimized Rust functions to improve performance and scalability for large datasets. The recommended way to use *mdrb* is by installing *metabodecon* and setting the backend argument to "rust" when calling its functions. The Rust part of the package is based on the metabodecon-rust crate.

⚠️ Attention: this package is experimental and its API is subject to change. When using in scripts and/or packages make sure to check the exact version first.

github/spang-lab/mdrb

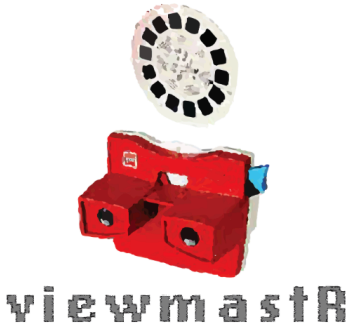
README MIT license

mspredictr



An R package for calculating peptide sequence and fragment masses relevant to mass spectrometry, along with some peptide string manipulation tools. Some of the functions with mass calculations and string manipulations have been written in `Rust` for a speed advantage.

github/jeffsocal/mspredictr



repo status Active lifecycle stable

viewmastR is a R framework for genomic cell type classification using the Burn machine learning library and its modules. viewmastR is a very flexible and customizable platform for labelling cell types in your data

The main features of viewmastR are:

- Use a blazingly fast machine learning approach to cell classification according to a reference dataset
- Augment data for rare cell types
- Classify single cell profiles according to a reference of bulk data

License

Full license

file LICENSE

Citation

Citing viewmastR

Developers

Scott Furlan

Author, maintainer

github/furlan-lab/viewmastR

extendr across-language

Color mode

Phylo2Vec

pypi package 1.3.1 | docs latest | DOI 10.5281/zenodo.15643504

SSEC Project | license LGPL-3.0

pre-commit.ci passed | CI Python passing | CI Rust passing | CI R passing

Phylo2Vec (or phylo2vec) is a high-performance software package for encoding, manipulating, and analysing binary phylogenetic trees. At its core, the package contains representation of binary trees, which defines a bijection from any tree topology with n leaves into an integer vector of size $n - 1$. Compared to the traditional Newick format, phylo2vec was designed with fast sampling, fast conversion/compression from Newick-format trees to the Phylo2Vec format, and rapid tree comparison in mind.

This current version features a core implementation in Rust, providing significant performance improvements and memory efficiency while remaining available in Python (superseding the version described in the original paper) and R via dedicated wrappers, making it accessible to a broad audience in the bioinformatics community.

Link to the paper: https://doi.org/10.1093/sysbio/syae030

`sbhattlab/phylo2vec/phylo2vec`

Exon

README License

Exon

Exon is an execution engine designed to work with bioinformatics data. It features:

- SQL based access to bioinformatics data -- general DML and some DDL support
- Support for many file formats from bioinformatics, proteomics, and others
- Local filesystem and object storage support
- Arrow FFI primitives for multi-language support

github/wheretrue/exon

and even more

- `CellBarcode`
- `taxozack`
- `rfaster2`
- `hrvhra`
- `fqtkWrapper`

Use case `extendr/mdl`

An example of a rust-powered R-package is {mdl}.

- Transforms a data-frame into a design/model matrix, that are used within lm/glm/glmnet/etc.

```
> mtcars$cyl <- as.factor(mtcars$cyl)
+ head(
+   mdl::mtrx(mpg ~ ., mtcars)
+ )
  (Intercept) cyl6 cyl8 disp  hp drat    wt  qsec vs am gear carb
1           1    1    0  160 110 3.90 2.620 16.46  0  1    4    4
2           1    1    0  160 110 3.90 2.875 17.02  0  1    4    4
3           1    0    0  108  93 3.85 2.320 18.61  1  1    4    1
4           1    1    0  258 110 3.08 3.215 19.44  1  0    3    1
5           1    0    1  360 175 3.15 3.440 17.02  0  0    3    2
6           1    1    0  225 105 2.76 3.460 20.22  1  0    3    1
```

Benchmark:


```
# A tibble: 2 × 4
  expression                          median `itr/sec` mem_alloc
  <bch:expr>                           <dbl>     <dbl>     <dbl>
1 mdl::mtrx(mpg ~ ., mtcars)             1        10.5      1
2 model.matrix(mpg ~ ., mtcars)         10.8       1        4.40
```

scaled wrt. best performing

Benchmark:


```
# A tibble: 2 × 4
  expression                          median `itr/sec` mem_alloc
  <bch:expr>                           <dbl>     <dbl>     <dbl>
1 mdl::mtrx(mpg ~ ., mtcars)             1        10.5      1
2 model.matrix(mpg ~ ., mtcars)         10.8       1        4.40
```

scaled wrt. best performing

Overall, between 1.7× and 11× faster than R's `model.matrix`.

But there is more.. Performance is not everything.

- The rust core of `mdl` is app. 250 LOC
- Parallel processing of variables is implemented (via `rayon`)
- 100% safe code

- The rust core of `mdl` is app. 250 LOC
- Parallel processing of variables is implemented (via `rayon`)
- 100% safe code

→ Any non-expert maintainer can tweak `mdl`, and if it compiles, it works.

Roadmap

Main priority: Developers, developers, developers

- Better support on package repositories like CRAN and Bioconductor
- Improve and complete our User Guide
- Outreach to users via presentations and workshops

More Rust in R's ecosystem

- Leverage existing rust crates through binding r-packages
- More maintainers for extendr

Thanks for your attention.

Thanks to Lluís Revilla for organising, hosting, and guiding us.

Let’s discuss!

Questions?

Maintainer burden

- New version of Rust (`rustc`/`cargo`) is released every 6 weeks

This is not an issue as

- `rustup` ensures that multiple compilers can be installed on the same machine, without conflict
- How does rust handle system dependencies?

Rust packages are called crates, and not *libraries*.

- `cargo` bundles all dependencies, thus no conflict can occur

Maintainer burden

- Rust's package repository is crates.io. Maintained by crates.io team and the Rust Foundation[1].

Previously, `cargo/crates.io` relied on an index registry hosted by GitHub. As of release 1.70.0, that is no longer the case.

- Rust guarantees **forward compatibility** of the different versions, however..
  - There are **Editions** which are opt-in breaking changes
  - There are new lints and warnings

Fear not, add Rust to your stack!

[1] Separate, legal entity, that funds the infrastructure work

cargo geiger mdl

```
Metric output format: x/y
    x = unsafe code used by the build
    y = total unsafe code found in the crate

Symbols:
    🔒  = No `unsafe` usage found, declares #![forbid(unsafe_code)]
    ?   = No `unsafe` usage found, missing #![forbid(unsafe_code)]
    ☢   = `unsafe` usage found

Functions  Expressions  Impls  Traits  Methods  Dependency
```

cargo geiger mdl

```
0/0        0/0          2/2    0/0    0/0      ☢  mdl 0.1.0
55/84      1990/3033    1/1    0/0    12/12    ☢  └── extendr-api 0.8.0
8/8        33/33        0/0    0/0    0/0      ☢      ├── extendr-ffi 0.8.0
0/0        0/0          0/1    0/0    0/0      ☢      ├── extendr-macros 0.8.0
0/0        15/15        0/0    0/0    3/3      ☢      │   ├── proc-macro2 1.0.86
0/0        4/4          0/0    0/0    0/0      ☢      │   │   └── unicode-ident
1.0.12
0/0        0/0          0/0    0/0    0/0      ?      │   ├── quote 1.0.37
0/0        15/15        0/0    0/0    3/3      ☢      │   │   └── proc-macro2 1.0.86
0/0        88/88        3/3    0/0    2/2      ☢      │   └── syn 2.0.77
0/0        15/15        0/0    0/0    3/3      ☢      │       ├── proc-macro2 1.0.86
0/0        0/0          0/0    0/0    0/0      ?      │       ├── quote 1.0.37
0/0        4/4          0/0    0/0    0/0      ☢      │       └── unicode-ident
1.0.12
0/0        81/124       5/9    0/0    3/5      ☢      ├── once_cell 1.21.3
0/0        0/0          0/0    0/0    0/0      ?      └── paste 1.0.15

63/92      2211/3297    11/16  0/0    20/22
```