

V and R

rvee: recreational v programming for R

Edwin de Jonge

Statistics Netherlands / UseR! 2022, @edwindjonge



Who am I?

Edwin de Jonge

- Statistical Consultant / R&D at Statistics Netherlands / CBS
- Author of some R packages (e.g. `whisker`, `docopt`):
<https://edwindj.r-universe.dev>
- Love R and like programming in general
- <http://github.com/edwindj>, @edwindjonge

Short summary

- V new simple programming language
- rvee : <https://github.com/edwindj/rvee> helps to create R extension in V.
- No runtime and R dependencies.
- Experimental, but works.

What is V?

- V is a programming language.

Simple, fast, safe, compiled. For developing maintainable software.

- Website: <https://vlang.io/>
- young, low-level,
- feels like scripting, but is strongly typed
- Very similar to Go, but “improves” on Go:
<https://vlang.io/compare#go>
- Very light weight

{.plain .fragile} Syntax example



```
fn main() {  
  areas := ['game', 'web', 'tools', 'science', 'systems',  
            'embedded', 'drivers', 'GUI', 'mobile']  
  for area in areas {  
    println('Hello, $area developers!')  
  }  
}
```

V is safe



- No null, undefined values: unexpected null causes software damage.
- **immutable** variables by default, making it easier to optimize and reason about code
- **Option**: functions returns ?Type and compiler generates error if empty return is not handled properly.
- **Result**: functions can return !Type and compiler generates error if run-time error is not handled properly.
- Other cases a function always returns a value
- Functions are pure by default (impurity is explicit)

V features

- Easy C-interop (C-backend)
- Fast compilation!
- Zero cost abstractions
- Sum Types
- Generics

Fast compilation

For comparison, space and time required to build each compiler:

| | Space | Build time |
|-------|------------------------|------------|
| Go | 525 MB | 1m 33s |
| Rust | 30 GB | 45m |
| GCC | 8 GB | 50m |
| Clang | 90 GB ^[0] | 60m |
| Swift | 70 GB ^[1] | 90m |
| V | < 10 MB ^[2] | <1s |

source: <https://vlang.io>

Performance (from V lang website)

- As fast as C (V's main backend compiles to human readable C)
- C interop without any costs
- Minimal amount of allocations
- Built-in serialization without runtime reflection
- Compiles to native binaries without any dependencies: a simple web server is only 65 KB

V has different compilation back-ends (like rust and Go)

- native code (includes a REPL)
- readable(!) C code
- javascript

R similarities

- V functions are *pure* by default, like R, arguments are immutable by default. Have to be marked `mutable`.
- V has `defer`, like R's `on.exit`
- Simple syntax, very flexible
- Has (some) reflection/code generation. (R beats that one)

- V has a module and pkg system, containing v src code (not compiled)
- Typical v-compilation results in one binary
- importing a module means including the v code.
- v can link though to C, CPP libraries.

What is rvee?

- An experimental R package to create R package extensions with functions written in V.
- Similar in concept to RCpp but for V.
- But no *compile-time* and *run-time* dependencies in resulting R package.
- (pkgname rv was already claimed for)

Why-o-why

- One of R's great powers is that of a glue language (just like S)
- V is a nice low-level language, so why not?
- Because it is fun :-)

Features rvee

Two possible routes for using v as extension for R: - Use v compiler to link to R.so / R.dll, but requires v to be present on CRAN. - Use v transpiler to C and create a “normal” R package. This is rvee

rvee

- Transpiles v code into C package code, including all wrappers
- include `vpkg:r` with wrapper functions for R's C API (30% complete, vector stuff works)

`rvee::rv_export_c`

- Put the v source in "`<pkg>/src/v`" directory
- decorate each function to be exported with `[rv_export]` attribute
- `rovee::rv_export_c("<pkg>")` generates:
 - "`./R/rv_export.R`": R functions calling the v functions declared in "`./src/v/rv_export.v`"
 - "`./src/v/rv_export.v`": v wrapper functions translating input and output to the original v functions.
 - "`./src/init.c`": registration code for the shared library
 - "`./src/.c`": the c code generated by v from the source files in the "`/src/v`" directory.
- And ready! No dependencies what so ever.
- After that `devtools::load_all` (or R CMD SHLIB) work.
(on the generated C files)

Other v_function

- `v_function` generates from `v` src code a working R function that calls that code

```
v_function('fn add(x int, y int, z int) int{  
    sum := x + y + z  
    return sum  
}')
```

```
add(1,2,3)
```

`vpkg r`

- wrapper library for R API.
- helps to transform V strings, arrays into R vectors (and vice versa).

Status rvee

- V is a young language, but already quite stable.
- Generated code is in C
- rvee is experimental,
- Help is welcome!
- Experimental, expect (breaking) changes!
- But. . . , hey it works :-)

Thank you!

Interested?

```
remotes::install_github("edwindj/rvee")
```

Or visit:

<http://github.com/edwindj/rvee>