Some sneakily cool features made it into the JuliaCall v0.17.2 CRAN release. With the latest version there is now an install_julia function for automatically installing Julia. This makes Julia a great high performance back end for R packages. For example, the following is an example from the diffeqr package that will work, even without Julia installed:

```
install.packages("diffeqr")
library(diffeqr)
de <- diffeqr::diffeq_setup()

lorenz <- function (u,p,t){
  du1 = p[1]*(u[2]-u[1])
  du2 = u[1]*(p[2]-u[3]) - u[2]
  du3 = u[1]*u[2] - p[3]*u[3]
  c(du1,du2,du3)
}
u0 <- c(1.0,1.0,1.0)
tspan <- c(0.0,100.0)
p <- c(10.0,28.0,8/3)
prob <- de$ODEProblem(lorenz,u0,tspan,p)
fastprob <- diffeqr::jitoptimize_ode(de,prob)
sol <- de$solve(fastprob,de$Tsit5(),saveat=0.01)
```

Under the hood it's using the DifferentialEquations.jl package and the SciML stack, but it's abstracted from users so much that Julia is essentially an alternative to Rcpp with easier interactive development. The following example really brings the seamless integration home:

```
install.packages(diffeqr)
library(diffeqr)
de <- diffeqr::diffeq_setup()
degpu <- diffeqr::diffeqgpu_setup()

lorenz <- function (u,p,t){
  du1 = p[1]*(u[2]-u[1])
  du2 = u[1]*(p[2]-u[3]) - u[2]
  du3 = u[1]*u[2] - p[3]*u[3]
  c(du1,du2,du3)
}
u0 <- c(1.0,1.0,1.0)
tspan <- c(0.0,100.0)
p <- c(10.0,28.0,8/3)
prob <- de$ODEProblem(lorenz,u0,tspan,p)
fastprob <- diffeqr::jitoptimize_ode(de,prob)

prob_func <- function (prob,i,rep){
  de$remake(prob,u0=runif(3)*u0,p=runif(3)*p)
}
ensembleprob = de$EnsembleProblem(fastprob, prob_func = prob_func,
safetycopy=FALSE)
sol <- de$solve(ensembleprob,de$Tsit5(),degpu$EnsembleGPUArray(),
trajectories=10000,saveat=0.01)
```

This example does the following:

1. Automatically installs Julia
2. Automatically installs DifferentialEquations.jl
3. Automatically installs CUDA (via CUDA.jl)
4. Automatically installs ModelingToolkit.jl and DiffEqGPU.jl
5. JIT transpiles the R function to Julia via ModelingToolkit
6. Uses KernelAbstractions (in DiffEqGPU) to generate a CUDA kernel from the Julia code
7. Solves the ODE 10,000 times with different parameters 350x over deSolve