



# EXTENDING R

## MOTIVATION, EXAMPLES, CONTEXT

---

Dirk Eddelbuettel

29 March 2018

Debian / R Project / U of Illinois

## Focus today is on

- R and its role
- Extensions
- The XR package
- Other approaches

# R AND ITS ROLE

---



From any one of

- csv
- txt
- xlsx
- xml, json, ...
- web scraping, ...
- hdf5, netcdf, ...
- sas, stata, spss, ...
- various SQL + NOSQL DBs
- various binary protocols

via

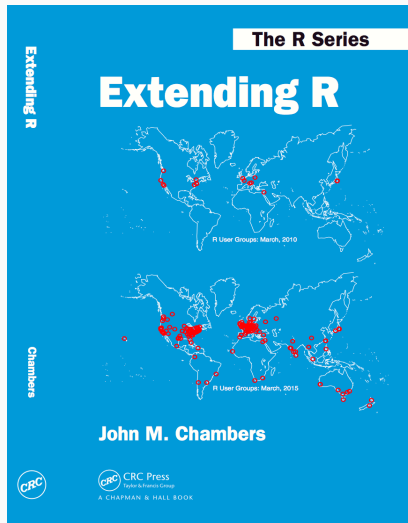


into any one of

- txt
- html
- latex and pdf
- html and js
- word
- shiny
- most graphics formats
- other dashboards
- web frontends

# R PER JOHN CHAMBERS (2016)

---



## Three Principles (Section 1.1)

Object Everything that exists in R is an object.

Function Everything that happens in R is a function call.

Interface Interfaces to other software are part of R.



## Three Principles (Section 1.1)

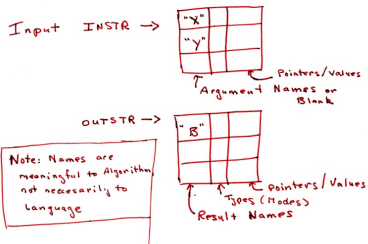
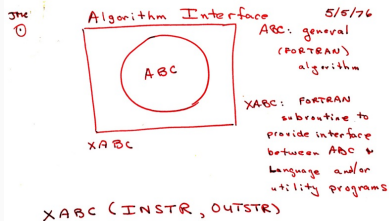
Object Everything that exists in R is an object.

Function Everything that happens in R is a function call.

Interface Interfaces to other software are part of R.

That is new. Or is it?

# R PER JOHN CHAMBERS (2016)



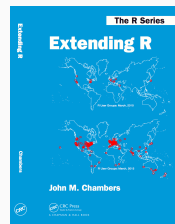
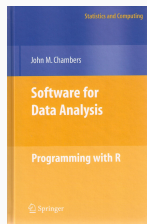
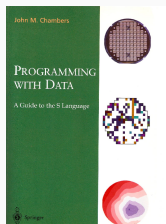
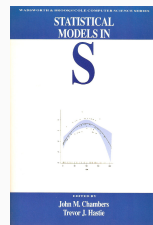
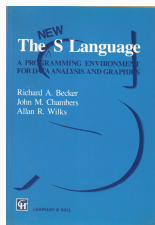
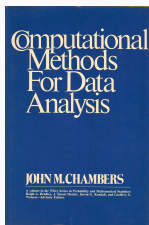
Source: John Chamber, personal communication

This became the “interface” system, a precursor to R.

# INTERLUDE

---

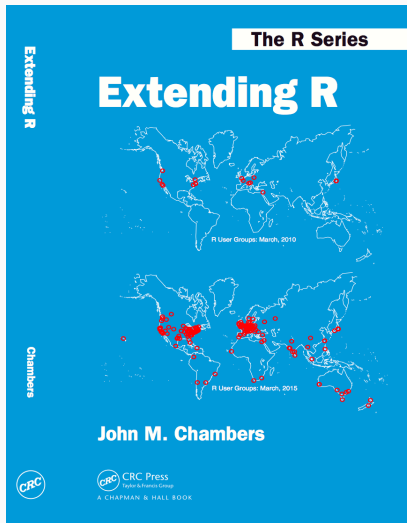
# PROGRAMMING WITH DATA FROM 1977 TO 2016



Thanks to John Chambers for high-resolution cover images. The publication years are, respectively, 1977, 1988, 1992, 1998, 2008 and 2016.

# XR







# XR BY JOHN CHAMBERS

## XR: A Structure for Interfaces from R

Support for interfaces from R to other languages, built around a class for evaluators and a combination of functions, classes and methods for communication. Will be used through a specific language interface package. Described in the book "Extending R".

Version: 0.7.2  
Imports: methods, utils, [jsonlite](#)  
Published: 2018-03-18  
Author: John M. Chambers  
Maintainer: John Chambers <jmc at r-project.org>  
License: [GPL-2](#) | [GPL-3](#) [expanded from: GPL (≥ 2)]  
NeedsCompilation: no  
In views: [NumericalMathematics](#)  
CRAN checks: [XR results](#)

### Downloads:

Reference manual: [XR.pdf](#)  
Package source: [XR\\_0.7.2.tar.gz](#)  
Windows binaries: r-prerelease: [XR\\_0.7.2.zip](#), r-release: [XR\\_0.7.2.zip](#), r-oldrel: [XR\\_0.7.2.zip](#)  
OS X binaries: r-prerelease: [XR\\_0.7.2.tgz](#), r-release: [XR\\_0.7.2.tgz](#)  
Old sources: [XR archive](#)

### Reverse dependencies:

Reverse imports: [XRJulia](#), [XRPython](#)

Source: <https://cloud.r-project.org/package=XR>

## XR: A framework for extending R

- Lower-level package
- Allows other (user-facing) packages to extend
- Examples of such application packages
  - XRPython
  - XRJulia
- No other languages bound *by this framework* AFAIK

## XR: A framework for extending R

- We will do a brief overview here
- Section 4 with 5 chapters has *all* the details
- XRPython and XRJulia contain their chapter as vignette
- Rcpp also has a full chapter (but no XRcpp package)
- rJava and others are mentioned

## XR: A framework for extending R

High-level view from 30,000 feet:

- XRPython uses *embedding* of an internal process
- XRJulia uses *socket* connection to external process
- (and Rcpp does what it does directly extending)

More discussion in *Extending R*.



About

Downloads

Documentation

**Python is powerful... and fast;  
plays well with others;  
runs everywhere;  
is friendly & easy to learn;  
is Open.**

These are some of the reasons people who use Python  
would rather not use anything else.

Source: <https://www.python.org/about/>

```
R> library(XRPython)
R> ev <- RPython()
R> ev
Python evaluator; \
Id: "Python Evaluator 2018-03-16 11:28:29.947906"; \
Evaluator number: 1
R>
```

```
ev$Eval(expr, ...)    # expression returning value
ev$Command(expr, ...) # statements

## equivalently
pythonEval(expr, ...)
pythonCommand(expr, ...)
```

## Simplest Example

```
R> library(XRPython)
R> ev <- RPython()
R> xx <- ev$Eval("[1, %s, 5]", pi)
R> ev$Command("print %s", xx)
```

```
[1, 3.14159265358979, 5]
```



## Shakespeare: Text Analysis Example

```
R> remotes::install_github("johnmchambers/shakespeare")
R> library(shakespeare) # also needs 'nlTK' + stopwords
R> hamlet <- parseXML(system.file("plays", "hamlet.xml",
                                package="shakespeare"))

R> hamlet
R Object of class "ElementTree_Python", \
for Python proxy object
Server Class: ElementTree; size: NA
R>
```

## Shakespeare: Text Analysis Example

```
R> hamlet$findtext("TITLE")
[1] "The Tragedy of Hamlet, Prince of Denmark"
R> speeches <- getSpeeches(hamlet)
R> speeches
R Object of class "SpeechList", for Python proxy object
Server Class: list; size: 1138
Field "tokens":
[1] TRUE
Field "tokenCase":
[1] FALSE
R>
```

## Shakespeare: Text Analysis Example

```
R> last <- speeches$pop()
R> pythonGet(last$getText())
[1] "Let four captains"
[2] "Bear Hamlet, like a soldier, to the stage;"
[3] "For he was likely, had he been put on,"
[4] "To have proved most royally: and, for his passage,"
[5] "The soldiers' music and the rites of war"
[6] "Speak loudly for him."
[7] "Take up the bodies: such a sight as this"
[8] "Becomes the field, but here shows much amiss."
[9] "Go, bid the soldiers shoot."
R>
```

Julia is a high-level, high-performance dynamic programming language for numerical computing. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library. Julia's Base library, largely written in Julia itself, also integrates mature, best-of-breed open source C and Fortran libraries for linear algebra, random number generation, signal processing, and string processing. In addition, the Julia developer community is contributing a number of external packages through Julia's built-in package manager at a rapid pace. IJulia, a collaboration between the Jupyter and Julia communities, provides a powerful browser-based graphical notebook interface to Julia.

Julia programs are organized around multiple dispatch, which allows built-in and user-defined functions to be overloaded for different combinations of argument types. For a more in-depth discussion of the rationale and advantages of Julia over other systems, see the following highlights or read the introduction in the online manual.

Source: <https://julialang.org/>

Installing Julia – using tarball into `/opt/julia`

```
edd@rob:~$ du -csh /opt/julia/  
230M    /opt/julia/  
230M    total  
edd@rob:~$
```

Once XRJulia is installed, on first use it downloads more into ~:

```
> library(XRJulia)
> ev <- RJulia()
Trying to add Julia package JSON; expect some messages and some delay
INFO: Initializing package repository /home/edd/.julia/v0.6
INFO: Cloning METADATA from https://github.com/JuliaLang/METADATA.jl
INFO: Cloning cache of Compat from https://github.com/JuliaLang/Compat.jl.git
INFO: Cloning cache of JSON from https://github.com/JuliaIO/JSON.jl.git
INFO: Cloning cache of Nullables from https://github.com/JuliaArchive/Nullables.jl.git
INFO: Installing Compat v0.60.0
INFO: Installing JSON v0.17.1
INFO: Installing Nullables v0.0.4
INFO: Package database updated
```

```
>
```

Here METADATA is 220 mb...

```
> library(XRJulia)
> set.seed(123)
> x <- matrix(rnorm(1000), 20, 5)
> xm <- juliaSend(x)
>
> xjm <- juliaGet(xm)
>
> all.equal(x, xjm)
[1] TRUE
```

```
> svdJ <- JuliaFunction("svdfact")
> sxm <- svdJ(xm)
> sxm
Julia proxy object
Server Class: Base.LinAlg.SVD{Float64,Float64,Array{Float64,2}}; size: NA
>
> sj <- juliaGet(sxm)
> sj@fields$S
[1] 4.75995 4.69669 4.32063 3.74208 2.43318
>
> sr <- svd(x) ## cf inst/tests/testSVD.R
> all.equal(sr$d, ep$Eval("%s.S",sj,.get=TRUE))
> all.equal(sr$u, ep$Eval("%s.U",sj,.get=TRUE))
> all.equal(t(sr$v), ep$Eval("%s.Vt",sj,.get=TRUE))
```



## OTHERS

---

## R Interface to Python

The **reticulate** package provides a comprehensive set of tools for interoperability between Python and R. The package includes facilities for:

- Translation between R and Python objects (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).
- Calling Python from R in a variety of ways including R Markdown, sourcing Python scripts, importing Python modules, and using Python interactively within an R session.
- Flexible binding to different versions of Python including virtual environments and Conda environments.



Reticulate embeds a Python session within your R session, enabling seamless, high-performance interoperability. If you are an R developer that uses Python for some of your work or a member of data science team that uses both languages, reticulate can dramatically streamline your workflow!

Source: <https://rstudio.github.io/reticulate/>

## reticulate

- Written to support **tensorflow** and **keras**
- Already used by several packages including
  - **greta**: think stan or bugs, but on tensorflow
  - **spacyr**: accesses the [spaCy](#) NLP engine
  - **h2o4gpu**: access to [h2o.ai](#) GPU-based ML solvers
- Also used by **XRPython**
- Uses Rcpp

The `RcppCNPY` package lets us load and save NumPy files (by wrapping the C library `cnpy`).

```
library(RcppCNPY)
mat <- npyLoad("fmat.npy")
vec <- npyLoad("fvec.npy")

mat2 <- npyLoad("fmat.npy.gz")
```

But `reticulate` lets us load and save NumPy files directly!

```
library(reticulate)
np <- import("numpy")
mat <- np$load("fmat.npy"))
vec <- np$load("fvec.npy"))

## compressed data: import gzip
gz <- import("gzip")
## use it to create handle to uncompressed file
mat2 <- np$load(gz$GzipFile("fmat.npy.gz", "r"))
```

## rJava

- oldest interface package
- fairly widely used
- mentioned in *Extending R*
- also:
  - `jdx` for data exchange
  - `js223` for Groovy, JavaScript, JRuby, Jython, Kotlin, ...
- `rscala` for di-rectional Scala interface

## Interface to Javascript

- V8 is a C++-based interface to Javascript
- large set of JS-based packages for browser-based work
- mentioned in *Extending R*
- Uses Rcpp

## Over 15 years old !

- still not widely known, yet used
- 'headless' R listening over tcp/ip
- by R Core member Simon Urbanek
- different client implementations



## RestRserve

[RestRserve](#) is an R web API framework for building **high-performance microservices and app backends**. The main difference with other frameworks ([plumber](#), [jug](#)) is that it is **parallel by design** (thanks to [Rserve](#)).

YES - it means it will handle all the incoming requests in parallel - each request in a separate fork.

### Features

- Create a http API by simply setting up a handler (R function) for a given route - [Hello-world](#)
- [Deploy applications](#) with a couple of lines of the code. Easily [stop](#) them.
- [Build high performance web API](#) - more than **20000 requests per second on a laptop** with 4 cores / 8 threads (Intel i7-7820HQ CPU), which is about **40x faster** than [plumber](#) (but of course these numbers are for illustration only - everything depends on the user code!).
- Generate [OpenAPI](#) specification by parsing annotations in R code
- Expose [Swagger UI](#)
- Serve static files

RestRserve is a very thin layer on the top of [Rserve](#) - most of the credits should go to [Simon Urbanek](#).

Source: <http://restrserve.org/>

## SOME OTHER THINGS

---

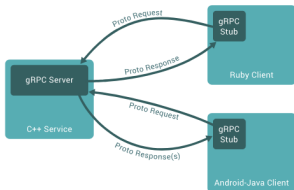
## RPy2

- Python package to call R from Python
- mature, well-tested
- but different direction

## Simple service definition

Define your service using Protocol Buffers, a powerful binary serialization toolset and language

READ MORE



## Works across languages and platforms

Automatically generate idiomatic client and server stubs for your service in a variety of languages and platforms

READ MORE

Source: <https://grpc.io>

## Different Approach

- define an *interface* (as Protocol Buffer)
- have code generated for both *server* and *client* side
- across OSs: Linux, Windows, Android, iOS, ...
- across languages: C++, Python, Go, Javascript, Ruby, C#, PHP, ...

# Apache Arrow

A cross-language development platform for in-memory data

Join Mailing List

Install (0.8.0 Release - 18 December 2017)

## See Latest News

Apache Arrow is a cross-language development platform for in-memory data. It specifies a standardized language-independent columnar memory format for flat and hierarchical data, organized for efficient analytic operations on modern hardware. It also provides computational libraries and zero-copy streaming messaging and interprocess communication. Languages currently supported include C, C++, Java, JavaScript, Python, and Ruby.

## Fast

Apache Arrow™ enables execution engines to take advantage of the latest SIMD (Single input multiple data) operations included in modern processors, for native vectorized optimization of analytical data processing. Columnar layout is optimized for data locality for better performance on modern hardware like CPUs and GPUs.

The Arrow memory format supports **zero-copy reads** for lightning-fast data access without serialization overhead.

## Flexible

Arrow acts as a new high-performance interface between various systems. It is also focused on supporting a wide variety of industry-standard programming languages. Java, C, C++, Python, Ruby, and JavaScript implementations are in progress and more languages are welcome.

## Standard

Apache Arrow is backed by key developers of 13 major open source projects, including Calcite, Cassandra, Drill, Hadoop, HBase, Ibis, Impala, Kudu, Pandas, Parquet, Phoenix, Spark, and Storm making it the de-facto standard for columnar in-memory analytics.

Learn more about projects that are [Powered By Apache Arrow](#)

Source: <https://arrow.apache.org/>



The C++ Tensor Algebra Library

Multi-dimensional arrays with broadcasting and lazy computing - all open-source.



Browse the Code



Documentation



Try it Now

## Introduction

**xtensor** is a C++ library meant for numerical analysis with multi-dimensional array expressions.

**xtensor** provides

- an extensible expression system enabling **lazy broadcasting**.
- an API following the idioms of the **C++ standard library**.
- tools to manipulate array expressions and build upon **xtensor**.

Source: <http://quantstack.net/xtensor>

# SUMMARY

---



## Interfaces from R

- are a natural part of the language!
- give us access to best of breed tools in other languages
- require some thought and care for design
- which *Extending R* discusses well

# THANK YOU!

slides <http://dirk.eddelbuettel.com/presentations/>

web <http://dirk.eddelbuettel.com/>

mail [dirk@eddelbuettel.com](mailto:dirk@eddelbuettel.com)

github [@eddelbuettel](#)

twitter [@eddelbuettel](#)