Extending R

Motivation, Examples, Context

Dirk Eddelbuettel

19 April 2018

Debian / R Project / U of Illinois

Overview

Outline

Focus today is on

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

R and its role

R as central point

R AS CENTRAL POINT

From any one of

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

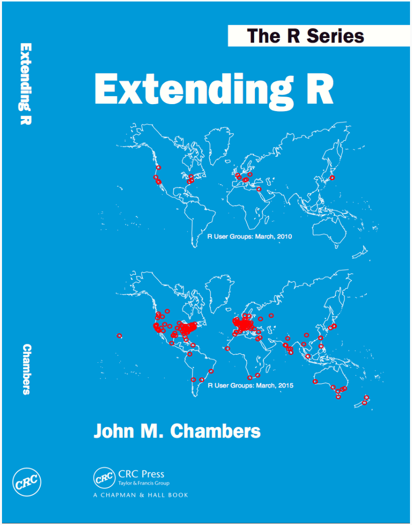
via

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)

R per John Chambers (2016)



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.

R per John Chambers (2016)

Three Principles (Section 1.1)

Object Everything that exists in R is an object.

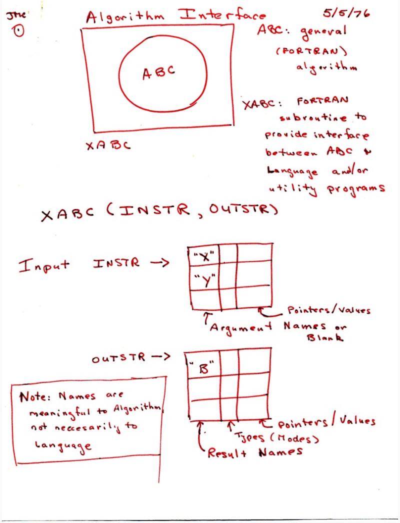
Function Everything that happens in R is a function call.

Interface Interfaces to other software are part of R.

R per John Chambers (2016)

That is new. Or is it?

R per John Chambers (2016)



JMC

1

Algorithm Interface

5/5/76

ABC

XABC

ABC: general (FORTRAN) algorithm

XABC: FORTRAN subroutine to provide interface between ABC & Language and/or utility programs

XABC (INSTR, OUTSTR)

Input INSTR ->

"X"

"Y"

Pointers/Values

Argument Names or Blank

OUTSTR ->

"B"

Pointers/Values

Types (Modes)

Result Names

Note: Names are meaningful to Algorithm, not neccesarily to language

Source: John Chamber, personal communication

Interface 1976

This became the system known as “Interface”, a precursor to S and R.

Interlude

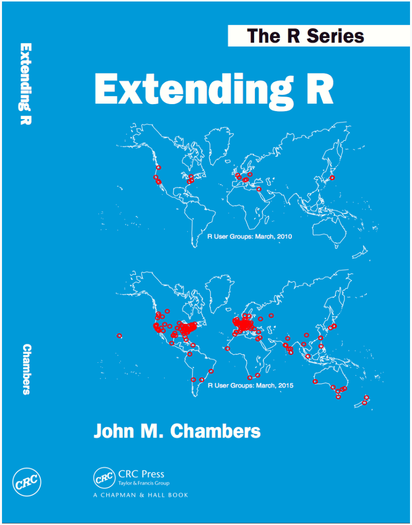
Programming with Data from 1977 to 2016

Computational Methods For Data Analysis
JOHN M.CHAMBERS
A volume in the Wiley Series in Probability and Mathematical Statistics: Ralph A. Bradley, J. Stuart Hunter, David G. Kendall, and Geoffrey S. Watson—Advisory Editors
The NEW S Language
A PROGRAMMING ENVIRONMENT FOR DATA ANALYSIS AND GRAPHICS
Richard A. Becker
John M. Chambers
Allan R. Wilks
CHAPMAN & HALL
WADSWORTH & BROOKS/COLE COMPUTER SCIENCE SERIES
STATISTICAL MODELS IN S
EDITED BY
John M. Chambers
Trevor J. Hastie
John M. Chambers
PROGRAMMING WITH DATA
A Guide to the S Language
Springer
Statistics and Computing
John M. Chambers
Software for Data Analysis
Programming with R
Springer
The R Series
Extending R
Extending R
Chambers
R User Groups: March, 2010
R User Groups: March, 2015
John M. Chambers
CRC Press
A CHAPMAN & HALL BOOK

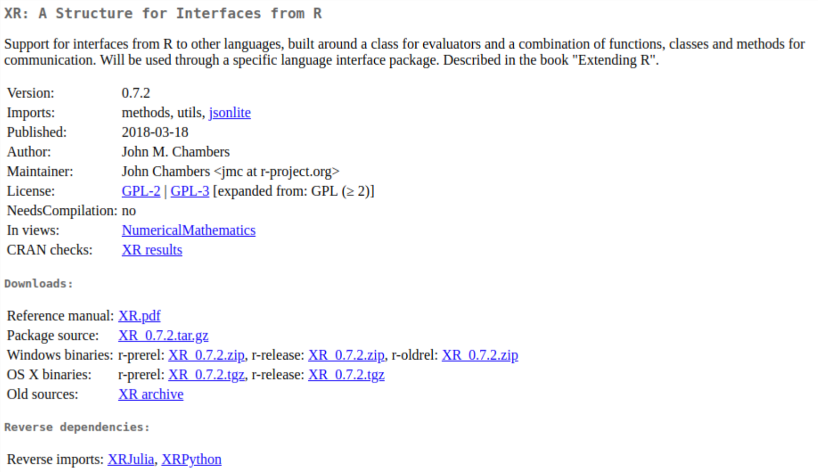
Thanks to John Chambers for high-resolution cover images. The publication years are, respectively, 1977, 1988, 1992, 1998, 2008 and 2016.

XR

XR by John Chambers



XR 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-prerel: XR_0.7.2.zip, r-release: XR_0.7.2.zip, r-oldrel: XR_0.7.2.zip
OS X binaries:	r-prerel: 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 by John Chambers

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 by John Chambers

XR: A framework for extending R

- We will do a brief overview here
- Section 4 with five 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 by John Chambers

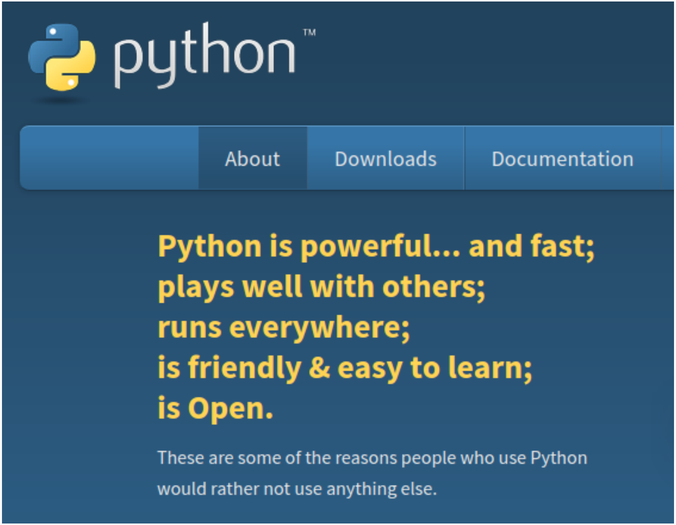
XR: A framework for extending R

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*.

XRPython



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/

XRPython

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

XRPython

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

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

XRPython

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]
```

XRPython

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>
```

XRPython

Shakespeare: Text Analysis Example

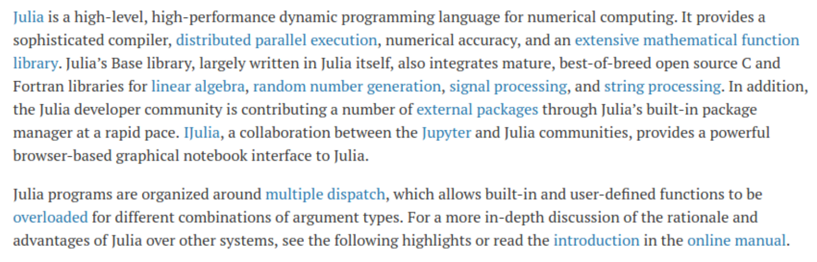
```
R> hamlet$findtext(”TITLE”)
[1] ”The Tragedy of Hamlet, Prince of Denmark”
R> speeches <- getSpeeches(hamlet)
R> speeches
R Object of class ”SpeechList”, for Python proxy object
Server Class: list; size: 1138
Field ”tokens”:
[1] TRUE
Field ”tokenCase”:
[1] FALSE
R>
```

XRPython

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>
```

XRJulia



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/

XRJulia

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

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

XRJulia

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…

XRJulia

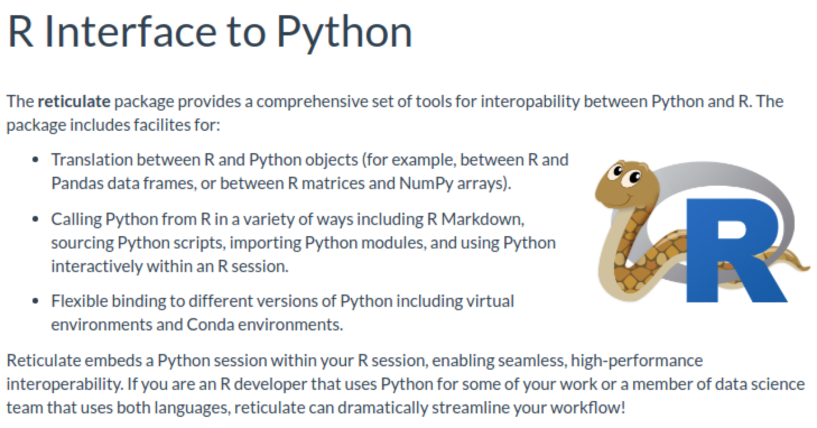
```
> library(XRJulia)
> set.seed(123)
> x <- matrix(rnorm(1000), 20, 5)
> xm <- juliaSend(x)
>
> xjm <- juliaGet(xm)
>
> all.equal(x, xjm)
[1] TRUE
```

XRJulia

```
> 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))
```

Alternatives

Generic Python wrapping



R Interface to Python

The **reticulate** package provides a comprehensive set of tools for interopability between Python and R. The package includes facilites 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/

Generic Python wrapping

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

Generic Python wrapping

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”)
```

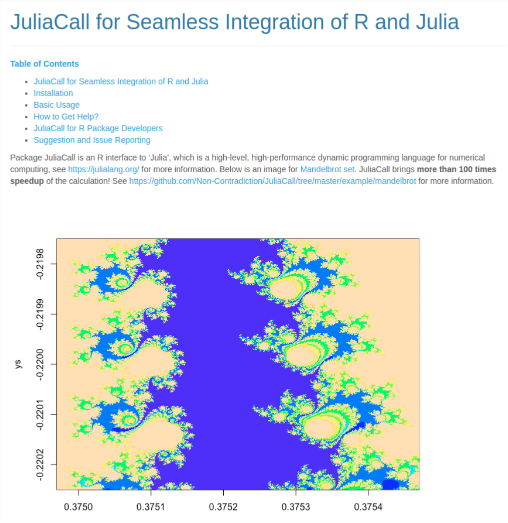
Generic Python wrapping

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”))
```

Generic Julia wrapping



JuliaCall for Seamless Integration of R and Julia

Table of Contents

Package JuliaCall is an R interface to 'Julia', which is a high-level, high-performance dynamic programming language for numerical computing, see https://julialang.org/ for more information. Below is an image for Mandelbrot set. JuliaCall brings **more than 100 times speedup** of the calculation! See https://github.com/Non-Contradiction/JuliaCall/tree/master/example/mandelbrot for more information.

Source: https://non-contradiction.github.io/JuliaCall/

Generic Julia wrapping

Similar initial setup issues as XRJulia

```r
library(JuliaCall)
jl <- julia_setup()
```

```
## Julia version 0.6.2 at location /opt/julia/bin will be used.
## Julia initiation...
## Finish Julia initiation.
## Loading setup script for JuliaCall...
## Finish loading setup script for JuliaCall.
```

(but it did blow up for me with RMarkdown)

Generic Julia wrapping

Different ways for using Julia to calculate sqrt(2)

```
julia_command(”a = sqrt(2);”); julia_eval(”a”)
# [1] 1.414214
julia_eval(”sqrt(2)”)
# [1] 1.414214
julia_call(”sqrt”, 2)
# [1] 1.414214
julia_eval(”sqrt”)(2)
#> [1] 1.414214
julia_assign(”x”, sqrt(2)); julia_eval(”x”)
#> [1] 1.414214
julia_assign(”rsqrt”, sqrt); julia_call(”rsqrt”, 2)
#> [1] 1.414214
2 %>J% sqrt
#> [1] 1.414214
```

Case Study

A Time-Tested Problem: Math

Consider a function defined as

$$\mathrm{f}(n) \quad \text{such that} \quad \begin{cases} n & \text{when } \ n < 2 \\ \mathrm{f}(n-1) + \mathrm{f}(n-2) & \text{when } \ n \geq 2 \end{cases}$$

A Time-Tested Problem: R

R implementation and use:

```
fr <- function(n) {
    if (n < 2) return(n)
    return(fr(n-1) + fr(n-2))
}

## Using it on first 11 arguments
sapply(0:10, fr)
```

```
##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

A Time-Tested Problem: R

Timing:

```r
library(rbenchmark)
benchmark(fr(10), fr(15), fr(20))[,1:4]
```

```
##     test replications elapsed relative
## 1 fr(10)          100   0.014    1.000
## 2 fr(15)          100   0.149   10.643
## 3 fr(20)          100   1.562  111.571
```

A Time-Tested Problem: Python

```python
def F(n):
    if n < 2: return n
    return F(n-1) + F(n-2)
```

saved as `fib.py`

A Time-Tested Problem: Python

We can use `reticulate`:

```r
## Python
library(reticulate)
py_run_file(”examples/fib.py”)
pr <- py_eval(”F(10)”)  # example call
fp <- function(n) py_eval(sprintf(”F(%d)”, n))

# Using it on first 11 arguments
sapply(0:10, fp)
```

```
##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

A Time-Tested Problem: Julia

In Julia, a one-liner will do

```
f(n) = n < 2 ? n : f(n-1) + f(n-2)
```

A Time-Tested Problem: Julia

We can use `JuliaCall`

```
library(JuliaCall)
fj <- julia_eval(”f(n) = n < 2 ? n : f(n-1) + f(n-2)”)
jr <- fj(10)

# Using it on first 11 arguments
sapply(0:10, fj)
```

```
## [1]  0  1  1  2  3  5  8 13 21 34 55
```

A Time-Tested Problem: C++

```cpp
int g(int n) {
    if (n < 2) return(n);
    return(g(n-1) + g(n-2));
}
```

A Time-Tested Problem: C++

Deployed *e.g.* as

```
Rcpp::cppFunction(”int fcpp(int n) {
   if (n < 2) return(n);
   return(fcpp(n-1) + fcpp(n-2)); }”)
sapply(0:10, fcpp)
```

```
##  [1]  0  1  1  2  3  5  8 13 21 34 55
```

A Time-Tested Problem: C++

Timing:

```
library(rbenchmark)
benchmark(fr(20), fp(20), fcpp(20),
          order=”relative”)[,1:4]
```

```
##        test replications elapsed relative
## 3 fcpp(20)          100   0.006    1.000
## 2   fp(20)          100   0.232   38.667
## 1   fr(20)          100   1.557  259.500
```

A nice gain of a few orders of magnitude.

A Time-Test Example: Four-ways

```
library(rbenchmark)

N <- 20
benchmark(fr(N), fp(N), fj(N), fcpp(N))[,1:4]
##      test replications elapsed relative
## 4 fcpp(N)          100   0.002      1.0
## 3   fj(N)          100   0.004      2.0
## 2   fp(N)          100   0.135     67.5
## 1   fr(N)          100   0.493    246.5
```

A Time-Test Example: Four-ways

```
N <- 30
benchmark(fr(N), fp(N), fj(N), fcpp(N))[,1:4]
##      test replications elapsed relative
## 4 fcpp(N)          100   0.207    1.000
## 3   fj(N)          100   0.365    1.763
## 2   fp(N)          100  17.110   82.657
## 1   fr(N)          100  61.662  297.884
## R>
```

Others

Java

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

V8

Interface to Javascript

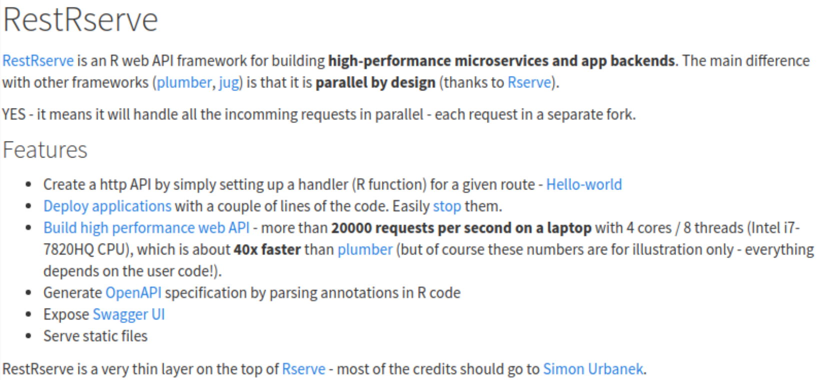
- V8 is a C++-based interface to Javascript
- large set of JS-based packages for browser-based work
- mentioned in *Extending R*
- Uses Rcpp

Rserve

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

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 incomming 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/

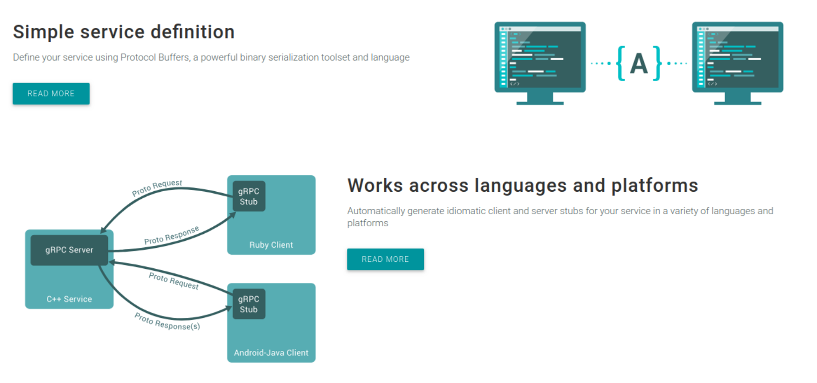
Some other things

Python the other way

RPy2

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

GRPC



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

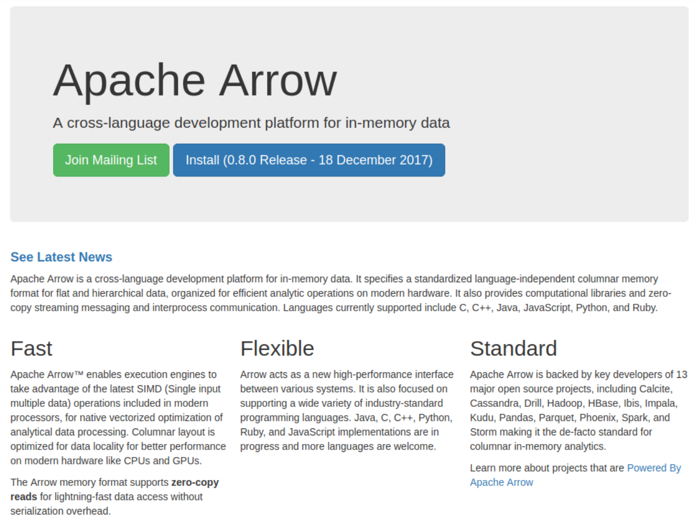
Source: https://grpc.io

gRPC

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, …

Arrow



Apache Arrow

A cross-language development platform for in-memory data

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.

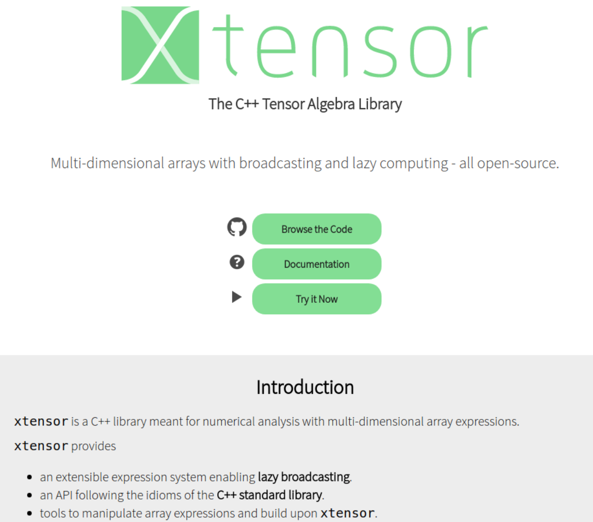
Standard

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/

XTENSOR



xtensor

The C++ Tensor Algebra Library

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

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

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
github	@eddelbuettel
twitter	@eddelbuettel