Polyglot applications with R and Python

[BARUG Meeting]

Laurent Gautier

DMAC / CBS

November 13th, 2012

Disclaimer

- This is not about:

Disclaimer

- This is not about:
    - The comparative merits of scripting languages

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R
- This is about:

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R
- This is about:
  - Accessing natively libraries implemented in a different language

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R
- This is about:
  - Accessing natively libraries implemented in a different language
  - Bridging people and skill sets through a glue language

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R
- This is about:
  - Accessing natively libraries implemented in a different language
  - Bridging people and skill sets through a glue language
  - Going from ideas to prototypes faster

Disclaimer

- This is not about:
  - The comparative merits of scripting languages
  - Only Python & R
- This is about:
  - Accessing natively libraries implemented in a different language
  - Bridging people and skill sets through a glue language
  - Going from ideas to prototypes faster
  - Putting some production into research

Disclaimer

- This is not about:
    - The comparative merits of scripting languages
    - Only Python & R
- This is about:
    - Accessing natively libraries implemented in a different language
    - Bridging people and skill sets through a glue language
    - Going from ideas to prototypes faster
    - Putting some production into research
    - Bringing research to production

Preamble

Scenario

- **data people:** Statisticians, data analysts

Preamble

Scenario

- **data people:** Statisticians, data analysts
- Data people have a method *M*

Preamble

Scenario

- **data people:** Statisticians, data analysts
- Data people have a method *M*
- Data people want to work on something new

Preamble

Scenario

- **data people:** Statisticians, data analysts
- Data people have a method *M*
- Data people want to work on something new
- Management wants an application for method *M*

Preamble

Scenario

- **data people:** Statisticians, data analysts
- Data people have a method *M*
- Data people want to work on something new
- Management wants an application for method *M*
- Management wants an application that uses method *M*

1 Only one language ?
- Polyglot programs

2 R and Python
- Mapping types
- Functions
- Evaluation and memory
- Building an application

Uniformity and coding standards

Uniformity and coding standards

Uniformity and coding standards

Get the job done

Get the job done

Soloist

Get the job done

Soloist

- Multi-talented individual
- Documentation ?

Get the job done

Soloist

- Multi-talented individual
- Documentation ?

Teamwork

Get the job done

Soloist

- Multi-talented individual
- Documentation ?

Teamwork

- Paired-programming
- Use the same tools ?
- Overlapping skills ?

Monolithic development

Monolithic development

- Centralized
- Top-down
- Lot of planning
- Long development, mostly only usable when complete
- Stand in time

Maintainability

Why use **one** unique language ?

- A legitimate managerial concern
- In places Java Certifications replaced general programming degrees
- Could good programmers matter more than the language ?
- Back to finding a needle in a haystack

Modularity at the heart of UNIX philosophy.

```
| > <
```

- No branching logic, unless going for shell scripts.
- Shell script no often thought after for applications
- The birth of scripting languages (Sed, Awk, Perl, . . .)

- Projects are cross-fields, cross-specialization
- Cost of specification - design - implementation too high
- Especially when the lifespan of the application is too short (or the user base too small).

Example from video games

- Engines (generally in C++)
- Scripting language for the ‘story’ and content
  - Python
  - Lua
  - Proprietary, others, . . .

- Large projects (with a lot of money at stake)
- Diverse competences (3D engine $\neq$ story logic)
- When speed of development is more important than speed of execution

This can apply to other industries

- Pipelines in visual effects
- Bioinformatics

1. Only one language ?
   - Polyglot programs

2. R and Python
   - Mapping types
   - Functions
   - Evaluation and memory
   - Building an application

R

- Language for statistics, data analysis, and data visualization
- Unmatched[1] number of libraries for anything having to do with data
- Specialized set of libraries for bioinformatics (Bioconductor)

[1] Almost certainly

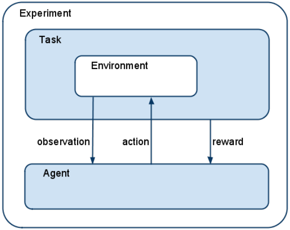
Python

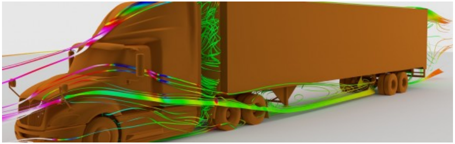
- All-purpose scripting language
- Unmatched[2] number of libraries for about anything
- Specialized sets of libraries for bioinformatics (Biopython, and a myriad smaller projects)

[2] May be

Python (continued)

- Machine learning R does not have: PyBrain
- Visualization tools R does not have: Mayavis, Blender


Experiment
Task
Environment
observation
action
reward
Agent



Python is popular in Bioinformatics / DNA sequencing.

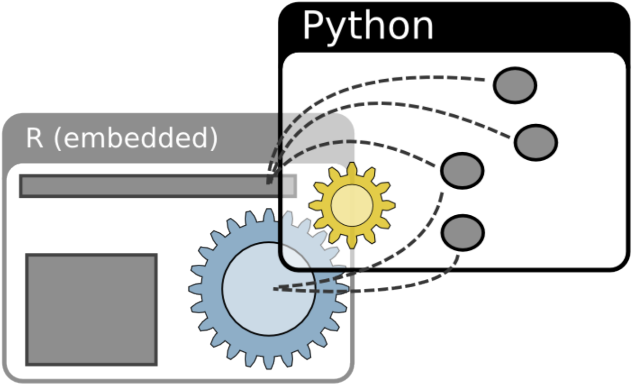
- Galaxy pipeline/server framework is in Python
- Some of the internal tools for the SOLiD are written in Python
- Ion Torrent Server is a Django server
- Oxford Nanopore control system is a server running Python

Why use anything else than R ?

- Build an application
- Work with very large data
- ‘Just because it can be done in R doesn’t mean you should do it’[3]

[3] John Dennison, R Meetup presentation

R embedded in Python



rpy2

- Feels like a regular Python library
- Embeds an R process
- Can be thought of as a stateful library

Two main parts:

- Low-level interface
- High-level interface

Low-level interface

- Close to R's C-API
- Let you do anything safe[4] from that API
- Expose R data structures as Python *builtin* structures

[4] or so is the intent

Types

R	rpy2	Python
numeric	**Float**SexpVector	float
integer	**Int**SexpVector	int
char	**Str**SexpVector	str
logical	**Bool**SexpVector	bool
complex	**Complex**SexpVector	complex
list	**List**SexpVector	list
environment	SexpEnvironment	dict
function	SexpClosure	function
S4	SexpS4	*object*
	SexpLang	*object*
	SexpExtPtr	*object*

Vectors and arrays

- C-like: Contiguous blocks of memory
- R objects exposed to Python as sequences or C-like arrays, with or without copy

R

```
    v <- seq(1, 10)
    v[1]   # select the first element
    w <- v + 1 # add 1 to all elts
```

R

```
    v <- seq(1, 10)
    v[1]   # select the first element
    w <- v + 1 # add 1 to all elts
```

rpy2.rinterface

```
import rpy2.rinterface as ri; ri.initr()
v = ri.IntSexpVector(range(1, 11))
v[0]   # select the first element
w = ri.IntSexpVector([x+1 for x in v])
```

R

```
    v <- seq(1, 10)
    v[1]   # select the first element
    w <- v + 1 # add 1 to all elts
```

rpy2.rinterface

```
import rpy2.rinterface as ri; ri.initr()
v = ri.IntSexpVector(range(1, 11))
v[0]   # select the first element
w = ri.IntSexpVector([x+1 for x in v])
```

rpy2.robjects

```
import rpy2.robjects as ro

v = ro.IntVector(range(1, 11))
v[0]   # select the first element
w = v.ro + 1
```

Missing values

NaN:

numeric data type value representing an undefined or unrepresentable value, especially in floating-point calculations.

- Also used for missing values.
- Is a standard.

NA:

- Used for missing values by R.
- Not a standard.

- Pitfall when passing data to C without copy/checks
- Applies to any C libraries (includes rpy2)

Functions

R functions can be called as if they were Python functions

```
import rpy2.robjects as ro

f = ro.r("function(x, y) { 2 * (x + y) }")

f(1, 2)
```

- conversion on-the-fly
- translated signatures (dot-to-underscore)

Packages and modules

R

Namespaces attached to the search path

```
> searchpaths()
[1] ".GlobalEnv"
[2] "/usr/local/packages/R/2.15/lib/R/library/stats"
[3] "/usr/local/packages/R/2.15/lib/R/library/graphics"
[4] "/usr/local/packages/R/2.15/lib/R/library/grDevices"
[5] "/usr/local/packages/R/2.15/lib/R/library/utils"
[6] "/usr/local/packages/R/2.15/lib/R/library/datasets"
[7] "/usr/local/packages/R/2.15/lib/R/library/methods"
[8] "Autoloads"
[9] "/usr/local/packages/R/2.15/lib/R/library/base"
```

Python

Python modules as namespaces

```
import os
os.path.basename('/path/to/a/file')
```

R packages (almost) as Python modules

```
from rpy2.robjects.packages import importr
stats = importr(’stats’)
# PCA !
pc = stats.prcomp(m)
```

R scripts as modules !

```
from rpy2.robjects.packages import SignatureTranslatedAnonymousPackage

# R code in a file as a package
with open(’rflib.R’) as f:
    code = ’’.join(f.readlines())
    rf = SignatureTranslatedAnonymousPackage(code, "rf")

imp = rf.get_importance(dataf, response)
```

R environments

- Associate symbols to objects
- Exposed as Python dictionaries (key - value)

R

```
env <- new.env()
assign(’x’, 123, envir = env)

y <- 456
```

Python

```
import rpy2.robjects as ro

env = ro.Environment()
env[’x’] = 123

ro.globalenv[’y’] = 456
```

R and callback functions

Common R idiom

```
# m: matrix of numerical values
f <- function(x) sum(x[x > 0])
res <- apply(m, 1, f)
```

How to do that with rpy2 ?

R and callback functions

Common R idiom

```
# m: matrix of numerical values
f <- function(x) sum(x[x > 0])
res <- apply(m, 1, f)
```

How to do that with rpy2 ?

```
import rpy2.interactive as r
import rpy2.rinterface as ri
r_code = """
  function(x)
    sum(x[x > 0])
"""
tmp = ri.parse(r_code)
eval = r.packages.base.eval
r_func = eval(tmp)
r.base.apply(m, 1, r_func)
```

R and callback functions

Common R idiom

```
# m: matrix of numerical values
f <- function(x) sum(x[x > 0])
res <- apply(m, 1, f)
```

How to do that with rpy2 ?

```
import rpy2.interactive as r
import rpy2.rinterface as ri
def tmp(x):
  gnr = elt for elt in x \
        if elt > 0
  return sum(gnr)
r_func = ri.rternalize(tmp)
r.base.apply(m, 1, r_func)
```

Evaluation strategies

R

- Pass-by-value / Call-by-value
- Modifying an object locally is always safe
- Unncessary copies

Python

- Pass-by-reference
- Explicit request if copy

Rpy2 exposes R as if it was pass-by-reference

Python

```python
from rpy2.robjects.vectors import IntVector

def f(x):
    x[0] = 123
v = ro.IntVector(range(1, 11))
f(v)
```

R

```r
f <- function(x) {
  x[0] = 123
  return(x)
}
v = seq(1, 11)
v = f(v)
```

Memory management and garbage collection

Memory management and garbage collection

R

- Tracing GC (check for reachability)
- *R_PreciousList*

Memory management and garbage collection

R

- Tracing GC (check for reachability)
- *R_PreciousList*

Python

- Reference counting
- Tracing GC

Memory management and garbage collection

R

- Tracing GC (check for reachability)
- *R_PreciousList*

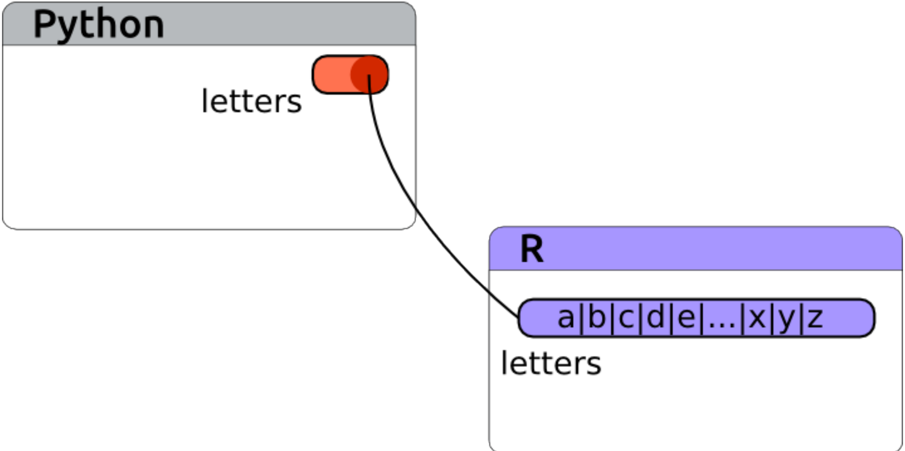
Python

- Reference counting
- Tracing GC

- Bridge different memory models
- Intermediate reference counting of R objects exposed
- That part could become very generic.

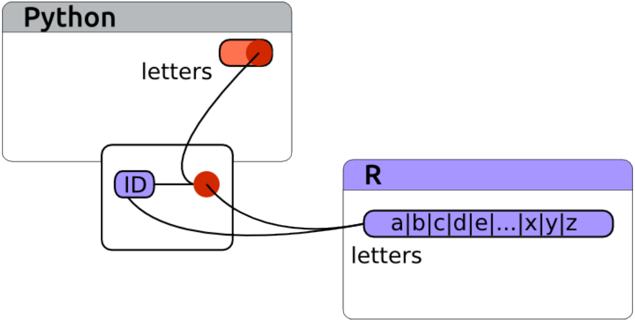
R objects exposed to R

```
import rpy2.rinterface as ri
ri.initr()
baseenv = ri.baseenv
letters = baseenv.get('letters')
```

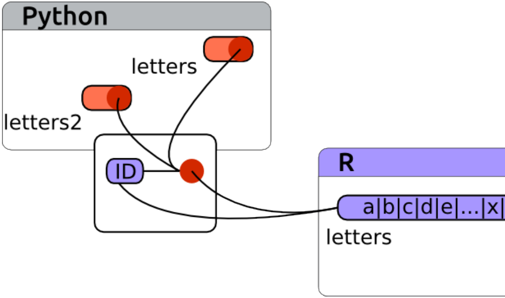


R objects exposed to R (not so simple)

```
import rpy2.rinterface as ri
ri.initr()
base = ri.baseenv
letters = base['letters']
```



```
>>> letters = base['letters']
>>> letters.rid # varies
123456
>>> letters.__sexp_refcount__
1
>>> letters2 = base['letters']
>>> letters2.__sexp_refcount__
2
>>> letters.__sexp_refcount__
2
>>> letters_2.rid # same R ID
123456
```



Exceptions

RRuntimeError: error while evaluating R code

KeyError: symbol not found in an environment

ValueError: invalid value passed to an rpy2 function

Performances

```
function(x) {
  total = 0;
  for (elt in x) {
    total <- total + elt
  }
}
```

Function	Sequence	Speedup
R		1.00
R compiled		6.52
R builtin		329.29
pure python	FloatVector	0.51
builtin python	FloatVector	0.54
pure python	SexpVector	7.45
builtin python	SexpVector	20.92
builtin python	array.array	53.62
builtin python	list	90.47

R through rpy2 can be faster than R

Let’s build a web application

- Why do that ?
  - Allow access to computing ressources
  - Use the UI of the browser
  - Good example
- Micro web framework: Flask

Hello world with Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

```
python hello.py
```

Importance of variables with random forest

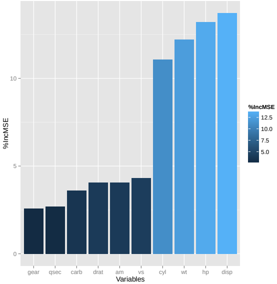
1. Data in a CSV file
2. Use R to compute a random forest and compute importance of variables
3. Make a pretty plot with *ggplot2*

```r
## data
dataf <- read.csv("/some/data/file.csv")
response <- 'var_name'

## importance of variables
library(randomForest)
get_importance <- function(dataf, response) {
  fmla <- formula(paste(response, '~ .'))
  dataf_rf <- randomForest(fmla, data = dataf,
                           keep.forest = FALSE,
                           importance = TRUE)
  imp <- importance(dataf_rf, type = 1)
  imp <- as.data.frame(imp[order(imp[,1]), , drop=FALSE])
  return(imp)
}

imp <- get_importance(dataf, response)

## plot
library(ggplot2)
get_plot <- function(imp) {
  rn <- rownames(imp)
  rn <- factor(rn, levels=rn, ordered=TRUE)
  imp <- cbind(as.data.frame(imp), rn = rn)
  p = ggplot(imp) +
    geom_bar(aes(y = `%IncMSE`,
                 x = rn,
                 fill = `%IncMSE`)) +
    scale_x_discrete("Variables")
  return(p)
}
p <- get_plot(imp)
print(p)
```



R library

```
get_dataframe <- function(filename) {
  return(read.csv(filename))
}

## importance of variables
library(randomForest)
get_importance <- function(dataf, response) {
  fmla <- formula(paste(response, '~ .'))
  dataf_rf <- randomForest(fmla, data = dataf,
                           keep.forest = FALSE,
                           importance = TRUE)
  imp <- importance(dataf_rf, type = 1)
  imp <- as.data.frame(imp[order(imp[,1]), , drop=FALSE])
  return(imp)
}
```

R library

```r
## plot
library(ggplot2)
get_plot <- function(imp) {
  rn <- rownames(imp)
  rn <- factor(rn, levels=rn, ordered=TRUE)
  imp <- cbind(as.data.frame(imp), rn = rn)
  p = ggplot(imp) +
    geom_bar(aes(y = `%IncMSE`,
                 x = rn,
                 fill = `%IncMSE`)) +
    scale_x_discrete("Variables")
  return(p)
}

make_PNGplot <- function(imp, dir) {
  filename <- tempfile(tmpdir = dir, fileext = '.png')
  p <- get_plot(imp)
  png(filename)
  print(p)
  dev.off()
  return(basename(filename))
}
```

Python application

```
import os
from flask import Flask, render_template, flash
from flask import url_for, send_from_directory
from flask import request
from werkzeug import secure_filename
from rpy2.robjects.packages import SignatureTranslatedAnonymousPackage

UPLOAD_FOLDER = ’/tmp’

# R code as a package
with open(’rflib.R’) as f:
    code = ’’.join(f.readlines())
    rf = SignatureTranslatedAnonymousPackage(code, "rf")
```

```python
# create application
app = Flask(__name__)
app.secret_key = 'change this !!!'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# serve files
@app.route('/files/<filename>')
def files(filename):
    return send_from_directory(UPLOAD_FOLDER,
                               filename)
```

```python
# create application
app = Flask(__name__)
app.secret_key = 'change this !!!'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

# serve files
@app.route('/files/<filename>')
def files(filename):
    return send_from_directory(UPLOAD_FOLDER,
                               filename)

def plot(dataf, response):
    # compute importance of variables
    imp = rf.get_importance(dataf, response)
    # plot into a file
    plot_fn = rf.make_PNGplot(imp, UPLOAD_FOLDER)[0]
    return url_for('files', filename = plot_fn)
```

```
# main function
@app.route(’/’, methods=[’GET’, ’POST’])
def index():
    plot_url = None
    # test if data posted
    if request.method == ’POST’:
        f = request.files[’data’]
        response = request.form[’response’]
        # test is file ’data’ uploaded
        if f:
            # save the uploaded file
            filename = secure_filename(f.filename)
            f.save(os.path.join(app.config[’UPLOAD_FOLDER’], filename))
            # get R data.frame from the file
            dataf = rf.get_dataframe(f.filename)
            # check if response variable is present
            if response in dataf.names:
                plot_url = plot(dataf, response)
            else:
                flash(’No such response variable’, category = ’error’)
        else:
            flash(’Invalid file extension’, category = ’error’)
    #
    return render_template(’index.html’, plot_url = plot_url)
```

Showtime…

Next steps

- Generic library to bridge R to anything with a C API
- Julia and R (hopefully end of 2012)