C.E.C.I

Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Introduction to Scripting Languages

damien.francois@uclouvain.be
October 2018

UCL Université catholique de Louvain

INSTITUT DE CALCUL INTENSIF ET DE STOCKAGE DE MASSE

Goal of this session:

“Advocate the use of scripting languages and help you choose the most suitable for your needs”

Agenda

1. Interpreters vs compilers
2. Octave, R, Python
3. GUIs & Literate programming
4. Packages/Libraries/Modules
5. When it is too slow
6. Bridges

Interpreters vs Compilers

C.E.C.I

- A **compiler** reads the whole code and produces a separate binary file that can be executed by the CPU.

  C/C++, Fortran, Java, Go, Haskel, ...

- An **interpreter** reads each line of code and executes it by calling the corresponding functionalities in its own code.

  Bash, Python, PHP, Javascript, Ruby, ...

Interpreters vs Compilers

- The ugly truth...
  - Many interpreters will pre-compile the code
  - Some compilers compile not to CPU-specific machine instructions but to bytecode
  - The bytecode interpreters sometimes re-compile the bytecode just before execution (JIT compiling)
  - Interpreters exist for C and C++
  - Compilers exist for Python
  - The interpreter can be compiled or himself interpreted

Interpreters vs Compilers

Compilers

- can apply code-wise powerful optimization
- practically have no run-time overhead

→ Speed

Interpreters

- allow easy code introspection
- offer high-level language constructs and tools

→ Ease of use

Interpreted languages

- Easier to **learn**
  - Many implementation details hidden
  - Can try and test code portions rapidly and easily
- Easier to **exchange**/reuse
  - The scripts are cross-platform by design
  - Often built-in package management
- Faster development
  - More **convenient programming** and shorter programs
    - Offers many simplifications and shortcuts – no need to micromanage memory
    - Built-in support for mundane tasks (handle files, dates, plots, NAs, NANs, ...)
  - **Easier to debug** and profile
    - GUI

Ex.1: argument parsing in Fortran

C.E.C.I

Parsing Command-Line Options in Fortran 2003

JASON BLEVINS
CV
RESEARCH
TEACHING
NOTES
TOOLS
LOG

ABOUT
ATOM FEED
TWITTER
CODE
GITHUB

SEPTEMBER 17, 2009

For programs with only a few simple command-line options, it isn't too difficult to parse them yourself, especially given Fortran 2003's new intrinsic functions command_argument_count and get_command_argument. Below is a simple example program which, by default, prints the current date and exits. It also accepts options to print the version, usage, or the current time. An error message is displayed if an invalid option is given.

```
! cmdline.f90 -- simple command-line argument parsing example

program cmdline
  implicit none

  character(len=*), parameter :: version = '1.0'
  character(len=32) :: arg
  character(len=8) :: date
  character(len=10) :: time
  character(len=5) :: zone
  logical :: do_time = .false.
  integer :: i

  do i = 1, command_argument_count()
     call get_command_argument(i, arg)

     select case (arg)
     case ('-v', '--version')
```


https://docs.python.org/3/library/argparse.html

Ex.1: argument parsing in Fortran

```
    call get_command_argument(i, arg)

    select case (arg)
    case ('-v', '--version')
       print '(2a)', 'cmdline version ', version
       stop
    case ('-h', '--help')
       call print_help()
       stop
    case ('-t', '--time')
       do_time = .true.
    case default
       print '(a,a,/)', 'Unrecognized command-line option: ', arg
       call print_help()
       stop
    end select
 end do

 ! Print the date and, optionally, the time
 call date_and_time(DATE=date, TIME=time, ZONE=zone)
 write (*, '(a,"-",a,"-",a)', advance='no') date(1:4), date(5:6), date(7:8)
 if (do_time) then
    write (*, '(x,a,":",a,x,a)') time(1:2), time(3:4), zone
 else
    write (*, '(a)') ''
 end if
```

Ex.1: argument parsing in Fortran

```
contains

  subroutine print_help()
    print '(a)', 'usage: cmdline [OPTIONS]'
    print '(a)', ''
    print '(a)', 'Without further options, cmdline prints the date and exits
    print '(a)', ''
    print '(a)', 'cmdline options:'
    print '(a)', ''
    print '(a)', '  -v, --version     print version information and exit'
    print '(a)', '  -h, --help        print usage information and exit'
    print '(a)', '  -t, --time        print time'
  end subroutine print_help

end program cmdline
```

Ex.1: argument parsing in Python

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                   help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                   const=sum, default=max,
                   help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

Assuming the Python code above is saved into a file called `prog.py`, it can be run at the command line and provides useful help messages:

```
$ python prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
 N           an integer for the accumulator

optional arguments:
 -h, --help  show this help message and exit
 --sum       sum the integers (default: find the max)
```


https://docs.python.org/3/library/argparse.html

Ex.2: Use XLS file in C

```c
                break;
            case 't':
                sheetName = strdup(optarg);
                break;
            case 'q':
                stringSeparator = optarg[0];
                break;
            case 'f':
                fieldSeparator = strdup(optarg);
                break;
            default:
                Usage(argv[0]);
                break;
            }
        }

            struct st_row_data* row;
            WORD cellRow, cellCol;

            // open workbook, choose standard conversion
            pWB = xls_open(argv[1], encoding);
            if (!pWB) {
                    fprintf(stderr, "File not found");
                    fprintf(stderr, "\n");
                    return EXIT_FAILURE;
            }

            // check if the requested sheet (if any) exists
            if (sheetName[0]) {
                    for (i = 0; i < pWB->sheets.count; i++) {
                            if (strcmp(sheetName, (char *)pWB->sheets.sheet[i].name) ==
                                    break;
                            }
                    }

                    if (i == pWB->sheets.count) {
                            fprintf(stderr, "Sheet \"%s\" not found", sheetName);
                            fprintf(stderr, "\n");
                            return EXIT_FAILURE;
                    }
            }

            // process all sheets
            for (i = 0; i < pWB->sheets.count; i++) {
                    int isFirstLine = 1;

            // just looking for sheet names
            if (justList) {
                printf("%s\n", pWB->sheets.sheet[i].name);
                continue;
            }

                    // check if this the sheet we want
                    if (sheetName[0]) {
                            if (strcmp(sheetName, (char *)pWB->sheets.sheet[i].name) !=
                                    continue;
                            }
                    }

                    // open and parse the sheet
                    pWS = xls_getWorkSheet(pWB, i);
                    xls_parseWorkSheet(pWS);

                    // process all rows of the sheet
                    for (cellRow = 0; cellRow <= pWS->rows.lastrow; cellRow++) {
                            int isFirstCol = 1;
                            row = xls_row(pWS, cellRow);

                            // process cells
                            if (!isFirstLine) {
                                    printf("%s", lineSeparator);
                            } else {
                                    isFirstLine = 0;
                            }

                            for (cellCol = 0; cellCol <= pWS->rows.lastcol; cellCol++) {
                    //printf("Processing row=%d col=%d\n", cellRow+1, cellCol+1);

                                    xlsCell *cell = xls_cell(pWS, cellRow, cellCol);
```

```c
                                    if ((!cell) || (cell->isHidden)) {
                                            continue;
                                    }

                                    if (!isFirstCol) {
                                            printf("%s", fieldSeparator);
                                    } else {
                                            isFirstCol = 0;
                                    }

                                    // display the colspan as only one cell, but reject
                                    if (cell->rowspan > 1) {
                                            fprintf(stderr, "Warning: %d rows spanned at
                                    }

                                    // display the value of the cell (either numeric or
                                    if (cell->id == 0x27e || cell->id == 0x0BD || cell->
                                            OutputNumber(cell->d);
                                    } else if (cell->id == 0x06) {
                            // formula
                                            if (cell->l == 0) // its a number
                                            {
                                                    OutputNumber(cell->d);
                                            } else {
                                                    if (!strcmp((char *)cell->str, "bool'
                                                    {
                                                            OutputString((int) cell->d ?
                                                    } else if (!strcmp((char *)cell->str
                                                    {
                                                            OutputString("*error*");
                                                    } else // ... cell->str is valid as
                                                    {
                                                            OutputString((char *)cell->s
                                                    }
                                            }
                                    } else if (cell->str != NULL) {
                                            OutputString((char *)cell->str);
                                    } else {
                                            OutputString("");
                                    }
                            }
                    }
                    xls_close_WS(pWS);
            }

            xls_close(pWB);
            return EXIT_SUCCESS;
}

// Output a CSV String (between double quotes)
// Escapes (doubles)" and \ characters
static void OutputString(const char *string) {
        const char *str;

        printf("%c", stringSeparator);
        for (str = string; *str; str++) {
                if (*str == stringSeparator) {
                        printf("%c%c", stringSeparator, stringSeparator);
                } else if (*str == '\\') {
                        printf("\\\\");
                } else {
                        printf("%c", *str);
                }
        }
        printf("%c", stringSeparator);
}

// Output a CSV Number
static void OutputNumber(const double number) {
        printf("%.15g", number);
}
```

Ex.2: Use XLS file in R

```
> mydata = read.xls("mydata.xls")  # read from first sheet
> write.csv(MyData, file = "MyData.csv")
```


https://cran.r-project.org/web/packages/gdata/

Ex.3: default args in Java

```
class DisplayOverloading
{
    public void disp(char c)
    {
         System.out.println(c);
    }
    public void disp(char c, int num)
    {
         System.out.println(c + " "+num);
    }
}
class Sample
{
   public static void main(String args[])
   {
       DisplayOverloading obj = new DisplayOverloading();
       obj.disp('a');
       obj.disp('a',10);
   }
}
```

Ex.3: default args in Octave

```
function hello (who = "World")
  printf ("Hello, %s!\n", who);
endfunction
```

When called without an input argument the function prints the following

```
hello ();
     -| Hello, World!
```

and when it's called with an input argument it prints the following

```
hello ("Beautiful World of Free Software");
     -| Hello, Beautiful World of Free Software!
```


https://www.gnu.org/software/octave/doc/interpreter/Default-Arguments.html

1.

Why those three?

Why those three?

- All very much used in scientific applications

    R (S/SPlus): strong for statistics

    Octave (Matlab): strong for engineering

    Python Scipy/Numpy (Canopy,Anaconda): strong for data science

- All free and free.
- Fun fact: All started as wrappers for Fortran code!

Why those three?

S was designed by John Chambers (Bell Lags) as an interactive interface to a Fortran-callable library, ca 1976.

MATLAB was built by Cleve Moler (University of New Mexico) to give students access to LINPACK and EISPACK without them having to learn Fortran

Python Numpy (Travis Oliphant, Brigham Young University) originates from f2py, a tool to easily extend Python with Fortran code.

Why those three?

Octave: Fortran optimized routines made easy to use. Easily handle (multi-dimensional) matrices, Nans, Infs, no need to worry about memory allocation, etc.

R: Easily handle matrices, strings, dates, and categories and missing values

Python: Full programming language, can handle custom objects

Why those three?

C.E.C.I

By contrast,

- Ruby, Perl: smaller bioinformatics-only community
- Javascript, PHP, Bash, TCL, Lua: totally different goal
- Matlab, IDL, Mathematica: not free
- Julia: very young – good luck to get help when needed

Why those three?

By contrast,

- Ruby, Perl: smaller bioinformatics-only community
- Javascript, PHP, Bash, TCL, Lua: totally different goal
- Matlab, IDL, Mathematica: not free
- Julia: very young – good luck to get help when needed

Not true anymore.
Worth considering !

(but not yet in this session...)

Why those three?

By contrast,

Ruby, Perl: smaller bioinformatics-only community

Javascript, PHP, Bash, TCL, Lua: totally different goal

Matlab, IDL, Mathematica: not free

Julia: very young – good luck to get help when needed

Not true anymore.
Worth considering !

(but not yet in this session…)

Some Julia in here...

2.

TripleQuickstart

Operators and assignment

a=1; b=2;	a=1; b=1	a<-1; b<-2
a + b	a + b or add(a,b)	a + b
a - b	a - b or subtract(a,b)	a - b
a * b	a * b or multiply(a,b)	a * b
a / b	a / b or divide(a,b)	a / b
a .^ b	a ** b	a ^ b
	power(a,b)	
	pow(a,b)	
rem(a,b)	a % b	a %% b
	remainder(a,b)	
	fmod(a,b)	
a(:,1) = 99	a[:,0] = 99	a[,1] <- 99
a(:,1) = [99 98 97]'	a[:,0] = array([99,98,97])	a[,1] <- c(99,98,97)
a(a>90) = 90;	(a>90).choose(a,90)	a[a>90] <- 90
	a.clip(min=None, max=90)	
	a.clip(min=2, max=5)	

http://sebastianraschka.com/Articles/2014_matrix_cheatsheet_table.html

Building arrays/matrices

`a=[2 3 4 5];`	`a=array([2,3,4,5])`	`a <- c(2,3,4,5)`
`adash=[2 3 4 5]';`	`array([2,3,4,5])[:,NewAxis]`	`adash <- t(c(2,3,4,5))`
	`array([2,3,4,5]).reshape(-1,1)`	
	`r_[1:10,'c']`	
`1:10`	`arange(1,11, dtype=Float)`	`seq(10)` or `1:10`
	`range(1,11)`	
`0:9`	`arange(10.)`	`seq(0,length=10)`
`1:3:10`	`arange(1,11,3)`	`seq(1,10,by=3)`
`10:-1:1`	`arange(10,0,-1)`	`seq(10,1)` or `10:1`
`10:-3:1`	`arange(10,0,-3)`	`seq(from=10,to=1,by=-3)`
`linspace(1,10,7)`	`linspace(1,10,7)`	`seq(1,10,length=7)`
`reverse(a)`	`a[::-1]` or	`rev(a)`
`a(:) = 3`	`a.fill(3), a[:] = 3`	

http://sebastianraschka.com/Articles/2014_matrix_cheatsheet_table.html

Indexing/slicing

`a(2,3)`	`a[1,2]`	`a[2,3]`
`a(1,:)`	`a[0,]`	`a[1,]`
`a(:,1)`	`a[:,0]`	`a[,1]`
`a([1 3],[1 4]);`	`a.take([0,2]).take([0,3], axis=1)`	
`a(2:end,:)`	`a[1:,]`	`a[-1,]`
`a(end-1:end,:)`	`a[-2:,]`	
`a(1:2:end,:)`	`a[::2,:]`	
	`a[...,2]`	
		`a[-2,-3]`
`a(:,[1 3 4])`	`a.take([0,2,3],axis=1)`	`a[,-2]`
	`a.diagonal(offset=0)`	

http://sebastianraschka.com/Articles/2014_matrix_cheatsheet_table.html

Searching arrays/matrices

`find(a)`	`a.ravel().nonzero()`	`which(a != 0)`
`[i j] = find(a)`	`(i,j) = a.nonzero()` `(i,j) = where(a!=0)`	`which(a != 0, arr.ind=T)`
`[i j v] = find(a)`	`v = a.compress((a!=0).flat)` `v = extract(a!=0,a)`	`ij <- which(a != 0, arr.ind=T); v <- a[ij]`
`find(a>5.5)`	`(a>5.5).nonzero()`	`which(a>5.5)`
	`a.compress((a>5.5).flat)`	`ij <- which(a>5.5, arr.ind=T); v <- a[ij]`
`a .* (a>5.5)`	`where(a>5.5,0,a) or a * (a>5.5)` `a.put(2,indices)`	

http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

Control structures

```
for i=1:5; disp(i); end
for i=1:5
   disp(i)
   disp(i*2)
end
```

MATLAB/Octave

```
if 1>0 a=100; end
if 1>0 a=100; else a=0; end
```

```
for i in range(1,6): print(i)
for i in range(1,6):
    print(i)
    print(i*2)
```

Python

```
if 1>0: a=100
```

```
for(i in 1:5) print(i)
for(i in 1:5) {
    print(i)
    print(i*2)
}
```

R

```
if (1>0) a <- 100

ifelse(a>0,a,0)
```


http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

More complete list

Hyperpolyglot

Numerical Analysis & Statistics: MATLAB, R, NumPy, Julia

a side-by-side reference sheet

sheet one: grammar and invocation | variables and expressions | arithmetic and logic | strings | regexes | dates and time | tuples | arrays | arithmetic sequences | 2d arrays | 3d arrays | dictionaries | functions | execution control | file handles | directories | processes and environment | libraries and namespaces | reflection | debugging

sheet two: tables | import and export | relational algebra | aggregation

vectors | matrices | sparse matrices | optimization | polynomials | descriptive statistics | distributions | linear regression | statistical tests | time series | fast fourier transform | clustering | images | sound

bar charts | scatter plots | line charts | surface charts | chart options

	matlab	r	numpy	julia
version used	*MATLAB 8.3* *Octave 3.8*	*3.1*	*Python 2.7* *NumPy 1.7* *SciPy 0.13* *Pandas 0.12* *Matplotlib 1.3*	*0.4*

show version	`$ matlab -nojvm -nodisplay -r 'exit'`
	`$ octave --version`
		`$ R --version`	`sys.version`
			`np.__version__`
			`sp.__version__`
			`mpl.__version__`
				`$ julia --version`

implicit prologue	*none*	`install.packages('ggplot2')`
		`library('ggplot2')`
			`import sys, os, re, math`
			`import numpy as np`
			`import scipy as sp`
			`import scipy.stats as stats`
			`import pandas as pd`
			`import matplotlib as mpl`
			`import matplotlib.pyplot as plt`	

		grammar and invocation		
	matlab	r	numpy	julia

interpreter	`$ cat >>foo.m`
	`1 + 1`
	`exit`
	`$ matlab -nojvm -nodisplay -r "run('foo.m')"`
	`$ octave foo.m`
		`$ cat >>foo.r`
		`1 + 1`
		`$ Rscript foo.r`
		`$ R -f foo.r`
			`$ cat >>foo.py`
			`print(1 + 1)`
			`$ python foo.py`
				`$ cat >>foo.jl`
				`println(1 + 1)`
				`$ julia foo.jl`

repl	`$ matlab -nojvm -nodisplay` `$ octave`	`$ R`	`$ python`	`$ julia`
command line program	`$ matlab -nojvm -nodisplay -r 'disp(1 + 1); exit'` `$ octave --silent --eval '1 + 1'`	`$ Rscript -e 'print("hi")'`	`python -c 'print("hi")'`	`$ julia -e 'println("hi")'`

block delimiters	`function end`
	`if elseif else end`
	`while end`
	`for end`
		`{ }`	*offside rule*	

http://hyperpolyglot.org/numerical-analysis

Linear regression

```
z = polyval(polyfit(x,y,1),x)
plot(x,y,'o', x,z ,'-')

a = x\y
```

```
(a,b) = polyfit(x,y,1)
plot(x,y,'o', x,a*x+b,'-')

linalg.lstsq(x,y)
```

```
z <- lm(y~x)
plot(x,y)
abline(z)
solve(a,b)
```


http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

Linear regression

```fortran
      SUBROUTINE MR (X, Y, N, K, DWORK, IWORK)
       IMPLICIT NONE
       INTEGER K, N, IWORK
       DOUBLE PRECISION X, Y, DWORK
       DIMENSION X(N,K), Y(N), DWORK(3*K), IWORK(K)

*         local variables
       INTEGER I, J
       DOUBLE PRECISION TAU, TOT

*        maximum of all column sums of magnitudes
       TAU = 0.
       DO J = 1, K
         TOT = 0.
         DO I = 1, N
           TOT = TOT + ABS(X(I,J))
         END DO
         IF (TOT > TAU) TAU = TOT
       END DO
       TAU = TAU * EPSILON(TAU)         ! tolerance argument

*            call function
       CALL DHFTI (X, N, N, K, Y, N, 1, TAU,
     $  J, DWORK(1), DWORK(K+1), DWORK(2*K+1), IWORK)
       IF (J < K) PRINT *, 'mr: solution is rank deficient!'
       RETURN
      END   ! of MR

*-----------------------------------------------------------------------
      PROGRAM t_mr         ! polynomial regression example
       IMPLICIT NONE
       INTEGER N, K
       PARAMETER (N=15, K=3)
       INTEGER IWORK(K), I, J
       DOUBLE PRECISION XIN(N), X(N,K), Y(N), DWORK(3*K)

       DATA XIN / 1.47, 1.50, 1.52, 1.55, 1.57, 1.60, 1.63, 1.65, 1.68,
     $            1.70, 1.73, 1.75, 1.78, 1.80, 1.83 /
       DATA Y / 52.21, 53.12, 54.48, 55.84, 57.20, 58.57, 59.93, 61.29,
     $          63.11, 64.47, 66.28, 68.10, 69.92, 72.19, 74.46 /

*            make coefficient matrix
       DO J = 1, K
         DO I = 1, N
           X(I,J) = XIN(I) **(J-1)
         END DO
       END DO

*             solve
       CALL MR (X, Y, N, K, DWORK, IWORK)

*             print result
  10   FORMAT ('beta: ', $)
  20   FORMAT (F12.4, $)
  30   FORMAT ()
       PRINT 10
       DO J = 1, K
         PRINT 20, Y(J)
       END DO
       PRINT 30
       STOP 'program complete'
      END
```

Fortran

```c
#include <stdio.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_math.h>
#include <gsl/gsl_multifit.h>

double w[] = {  52.21, 53.12, 54.48, 55.84, 57.20,
                58.57, 59.93, 61.29, 63.11, 64.47,
                66.28, 68.10, 69.92, 72.19, 74.46 };
double h[] = {  1.47, 1.50, 1.52, 1.55, 1.57,
                1.60, 1.63, 1.65, 1.68, 1.70,
                1.73, 1.75, 1.78, 1.80, 1.83    };

int main()
{
        int n = sizeof(h)/sizeof(double);
        gsl_matrix *X = gsl_matrix_calloc(n, 3);
        gsl_vector *Y = gsl_vector_alloc(n);
        gsl_vector *beta = gsl_vector_alloc(3);

        for (int i = 0; i < n; i++) {
                gsl_vector_set(Y, i, w[i]);

                gsl_matrix_set(X, i, 0, 1);
                gsl_matrix_set(X, i, 1, h[i]);
                gsl_matrix_set(X, i, 2, h[i] * h[i]);
        }

        double chisq;
        gsl_matrix *cov = gsl_matrix_alloc(3, 3);
        gsl_multifit_linear_workspace * wspc = gsl_multifit_linear_alloc(n, 3);
        gsl_multifit_linear(X, Y, beta, cov, &chisq, wspc);

        printf("Beta:");
        for (int i = 0; i < 3; i++)
                printf("  %g", gsl_vector_get(beta, i));
        printf("\n");

        gsl_matrix_free(X);
        gsl_matrix_free(cov);
        gsl_vector_free(Y);
        gsl_vector_free(beta);
        gsl_multifit_linear_free(wspc);

}
```

C

http://mathesaurus.sourceforge.net/matlab-python-xref.pdf

So..

Fast to learn
Fast to code

Challenge.. Write 'sapin.[m|R|py]'

dfr@hmem00 — bash

```
dfr@hmem00:~/scripting $ octave -q sapin.m 5 3
    #
   #@#
  #@##@
 ##@##@#
#@##@##@#
```

[redacted]

```
dfr@hmem00:~/scripting $ octave -q sapin.m 10 3
         #
        #@#
       #@##@
      ##@##@#
     #@##@##@#
    #@##@##@##@
   ##@##@##@##@#
  #@##@##@##@##@#
 #@##@##@##@##@##@
##@##@##@##@##@##@#
```

[redacted]

dfr@hmem00 — bash

```
dfr@hmem00:~/scripting $ octave -q sapin.m 10 6
         #
        ###
       #@###
      ##@####
     #@#####@#
    ####@#####@
   #####@#####@#
  ####@#####@#####
 #@#####@#####@###
##@#####@#####@####
```

[redacted]

```
dfr@hmem00:~/scripting $
```

Challenge.. Write 'sapin.[m|R|py]'

dfr@hmem00 — bash

```
dfr@hmem00:~/scripting $ Rscript sapin.R
          #
         ###
        #@###
       ##@####
      #@#####@#
     ####@######@
    #####@######@#
   ####@######@#####
  #@#####@######@###
 ##@######@######@####
#@#####@#####@######@#
dfr@hmem00:~/scripting $
```

dfr@hmem00 — bash

```
dfr@hmem00:~/scripting $ python sapin.py
         #
        ###
       #@###
      ##@####
     #@#####@#
    ####@######@
   #####@######@#
  ####@######@#####
 #@#####@######@###
##@######@######@####
dfr@hmem00:~/scripting $
```

Help

You will need for-loops, if-conditionals, variable assignment, and printing which you can find in the slides

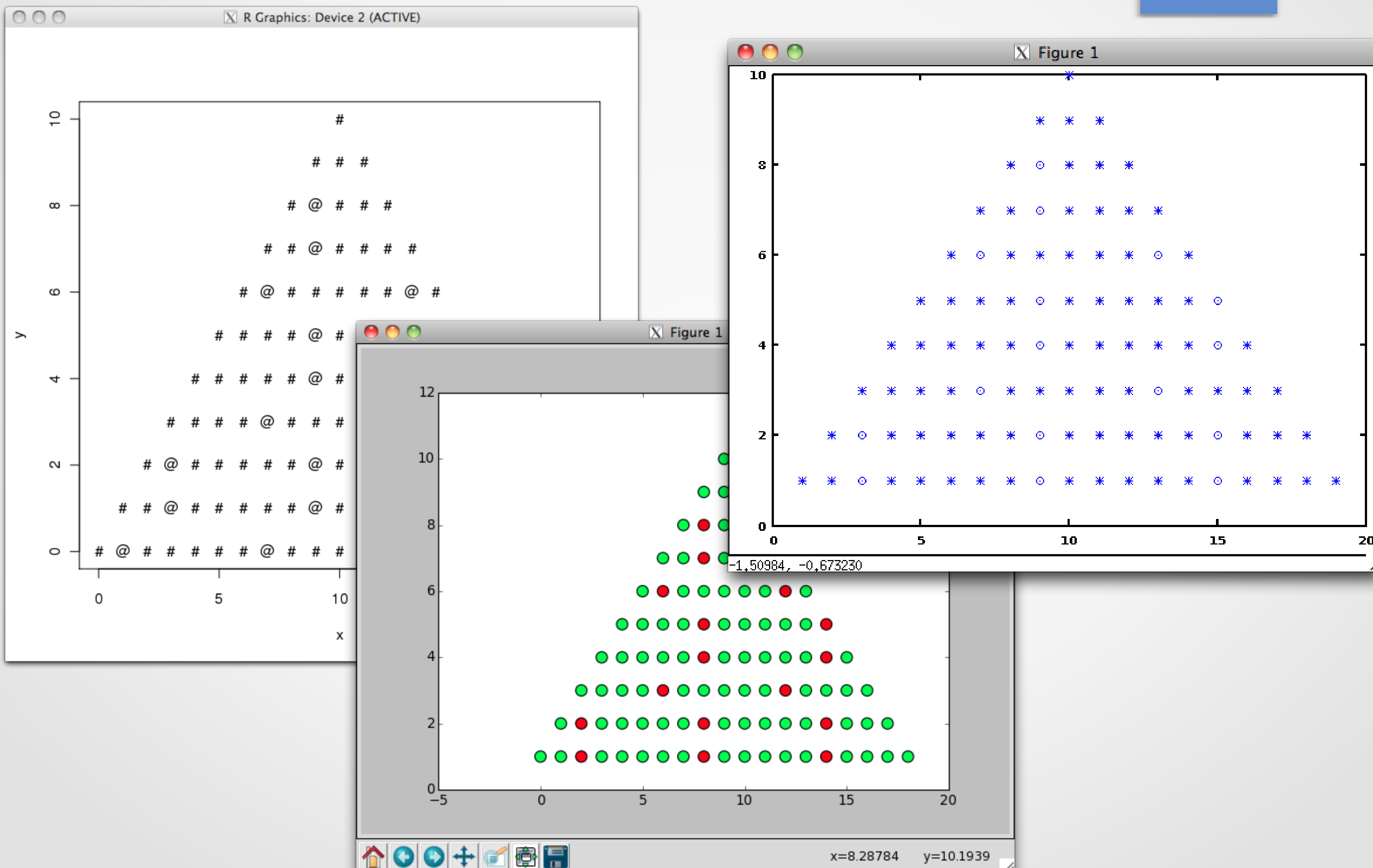
Other resources:
https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Getting_started
https://cran.r-project.org/doc/manuals/R-intro.html
http://wiki.scipy.org/Tentative_NumPy_Tutorial

http://stackoverflow.com/questions/14395569/how-to-output-text-in-the-r-console-without-creating-new-lines

http://stackoverflow.com/questions/493386/how-to-print-in-python-without-newline-or-space

http://stackoverflow.com/questions/1012597/displaying-information-from-matlab-without-a-line-feed

If you are that quick... Try this:



Possible solution (C)

dfr@hmem00 — bash

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int h=10;
int p=6;

int i, j, c=0;
char pat[] = "#@";

void usage()
{
    printf("usage: sapin.m [-h] [n [p]]\n"
           "\n"
           "Prints a christmas tree\n"
           "\n"
           "optional arguments:\n"
           "  -h show this help message and exit\n"
           "  n  Tree height\n"
           "  p  Decoration period\n");
    exit(1);
}


int main(int argc, char **argv)
{
    if (argc == 2 && !strcmp(argv[1], "-h"))
        usage();

```

1,1 Top

Possible solution (C, cont'd)

dfr@hmem00 — bash

```c
            "optional arguments:\n"
            "   -h show this help message and exit\n"
            "   n  Tree height\n"
            "   p  Decoration period\n");
    exit(1);
}


int main(int argc, char **argv)
{
    if (argc == 2 && !strcmp(argv[1], "-h"))
        usage();

    if (argc>1)
        h = atoi(argv[1]);

    if (argc>2)
        p = atoi(argv[2]);

    for (i=1; i<=h; i++)
    {
        for (j=0; j<h-i; j++)
            printf(" ");
        for (j=0; j< 2*i-1; j++)
            printf("%c", pat[!(++c%p)]);
        printf("\n");
    }
    return 0;
}
```

45,1 Bot

Possible solution (Octave)

dfr@hmem00 — bash

```
if nargin ==1 && argv(){1} == '-h'
    disp('usage: sapin.m [-h] [n [p]]')
    disp('')
    disp('Prints a christmas tree')
    disp('')
    disp('optional arguments:')
    disp('  -h show this help message and exit')
    disp('  n  Tree height')
    disp('  p  Decoration period')
    exit
end

if nargin > 0
    h=str2num(argv(){1});
else
    h=10;
end

if nargin > 1
    p=str2num(argv(){2});
else
    p=6;
end

for i = 0:h
    line = repmat('#', 1, 2*i + 1);
    line(p-mod((i)^2, p):p:end)='@';
    printf('%s%s\n', repmat(' ', 1, h-i), line)
end
```

:   29,1   All

Possible solution (R)

dfr@hmem00 — bash

```
opts <- commandArgs(trailingOnly=TRUE)
if (length(opts) == 1 & opts[1] == '-h') {
    cat('usage: sapin.m [-h] [n [p]]\n\n')
    cat('Prints a christmas tree\n\n')
    cat('optional arguments:\n')
    cat('  -h show this help message and exit\n')
    cat('  n  Tree height\n')
    cat('  p  Decoration period\n')
    q()
}

if (length(opts) > 0) {
    h <- as.numeric(opts[1])
} else {
    h <- 10
}
if (length(opts) > 1) {
    p <- as.numeric(opts[2])
} else {
    p <- 6
}

lst <- rep(c(rep('#', p-1), '@'), (h*h+1))

for (i in 0:h ) {
    top <- head(lst, 2*i+1)
    lst <- tail(lst, -(2*i+1))
    cat(paste(c(rep(' ', h - i ), top), sep="", collapse=""), '\n')
}
"sapin.R" 29L, 671C written                              1,1          All
```

Possible solution (Python)

dfr@hmem00 — bash

```
#! /bin/env python

from argparse import ArgumentParser
from itertools import cycle, islice

argparser = ArgumentParser(description='Prints a christmas tree')
argparser.add_argument('-n', dest='h', help='Tree height', default=10, type=int)
argparser.add_argument('-p', dest='p', help='Decoration period', default=6, type=int)

args = argparser.parse_args()

c = cycle('#' * (args.p - 1) + '@')

for i in xrange(args.h):
    print ' ' * (args.h - i - 1) + ''.join(list(islice(c, i * 2 + 1)))
~
~
~
~
~
~
~
~
~
~
~
~
:set wrap                                                    7,1           All
```

Possible solution (Julia)

9 - sapin

```
using ArgParse
using Parameters

s = ArgParseSettings()

@add_arg_table s begin
    "-n"
        help = "Tree height"
        arg_type = Int
        default = 10
    "-p"
        help = "Decoration period"
        arg_type = Int
        default = 6
end

@unpack n, p = parse_args(s)

function print_tree(height, count=0)
    height == 0 && return count
    count = print_tree(height-1, count)
    width = 2*height - 1
    offset = p-mod(count, p)-1
    print(' '^(n-height))
    println(("#" ^ offset * ('@' * '#'^(p-1))^(1+div(width,p)))[1:width])
    return count += width
end

print_tree(n)
~
```

N… sapin.jl jul… 3% ☰ 1: 1

Second challenge

```
dfr@lemaitre2:/CECI/home/ucl/pan/dfr/scripting/resmerge $ ls *txt
res-10.txt  res-24.txt  res-38.txt  res-51.txt  res-65.txt  res-79.txt  res-92.txt
res-11.txt  res-25.txt  res-39.txt  res-52.txt  res-66.txt  res-7.txt   res-93.txt
res-12.txt  res-26.txt  res-3.txt   res-53.txt  res-67.txt  res-80.txt  res-94.txt
res-13.txt  res-27.txt  res-40.txt  res-54.txt  res-68.txt  res-81.txt  res-95.txt
res-14.txt  res-28.txt  res-41.txt  res-55.txt  res-69.txt  res-82.txt  res-96.txt
res-15.txt  res-29.txt  res-42.txt  res-56.txt  res-6.txt   res-83.txt  res-97.txt
res-16.txt  res-2.txt   res-43.txt  res-57.txt  res-70.txt  res-84.txt  res-98.txt
res-17.txt  res-30.txt  res-44.txt  res-58.txt  res-71.txt  res-85.txt  res-99.txt
res-18.txt  res-31.txt  res-45.txt  res-59.txt  res-72.txt  res-86.txt  res-9.txt
res-19.txt  res-32.txt  res-46.txt  res-5.txt   res-73.txt  res-87.txt
res-1.txt   res-33.txt  res-47.txt  res-60.txt  res-74.txt  res-88.txt
res-20.txt  res-34.txt  res-48.txt  res-61.txt  res-75.txt  res-89.txt
res-21.txt  res-35.txt  res-49.txt  res-62.txt  res-76.txt  res-8.txt
res-22.txt  res-36.txt  res-4.txt   res-63.txt  res-77.txt  res-90.txt
res-23.txt  res-37.txt  res-50.txt  res-64.txt  res-78.txt  res-91.txt
dfr@lemaitre2:/CECI/home/ucl/pan/dfr/scripting/resmerge $ cat res-1.txt
# Result file for experiment
[main]

parameter=0.01
result=0.15492

[meta]
time=531244
```

Second challenge

- Find for which value of 'parameter' is 'result' the lowest.
- Course of action:
  - Read all files and parse them (you might need to install additional packages/libraries/modules)
  - Build two arrays one of parameter values and the other one for result values
  - Remove problematic values (plotting might help here)
  - Find minimum

Possible solution

```
nb_res=99;

p=zeros(nb_res,1);
r=zeros(nb_res,1);

for i = 1:nb_res;
  res = ini2struct(sprintf("res-%d.txt", i));
  p(i)=str2double(res.main.parameter);
  r(i)=str2double(res.main.result);
end
r(diff(r)>0.1)=nan;
plot(p,r)
[i, j]=min(r);
i, p(j)
~
~
~
~
~
~
```

```
library(ini)

nb_res <-99

p <- numeric(nb_res)
r <- numeric(nb_res)

for (i in 1:nb_res) {
  f <- read.ini(sprintf('res-%d.txt', i))
  p[i] <- as.numeric(f$main$parameter )
  r[i] <- as.numeric(f$main$result )
}

plot(p,r, 'l')
r[diff(r) > 0.1] <- NA
print(min(r, na.rm=T))
print(p[which.min(r)])
```

```
import configparser
import numpy as np
import matplotlib.pyplot as plt

nb_res = 99

p = np.zeros(nb_res)
r = np.zeros(nb_res)

for i in range(nb_res):
    f = configparser.RawConfigParser()
    f.read("res-{i}.txt".format(i=i+1))
    p[i] = float(f.get('main', 'parameter'))
    r[i] = float(f.get('main', 'result'))


plt.plot(p, r, '-')
r[np.where(np.diff(r) > .1)] = np.nan
print(np.nanmin(r))
print(p[np.nanargmin(r)])
```

- https://nl.mathworks.com/matlabcentral/fileexchange/17177-ini2struct
- https://cran.r-project.org/web/packages/ini/index.html
- https://docs.python.org/3/library/configparser.html

Possible solution

C.E.C.I

julia

9 - resmerge

```
using IniFile
using Plots

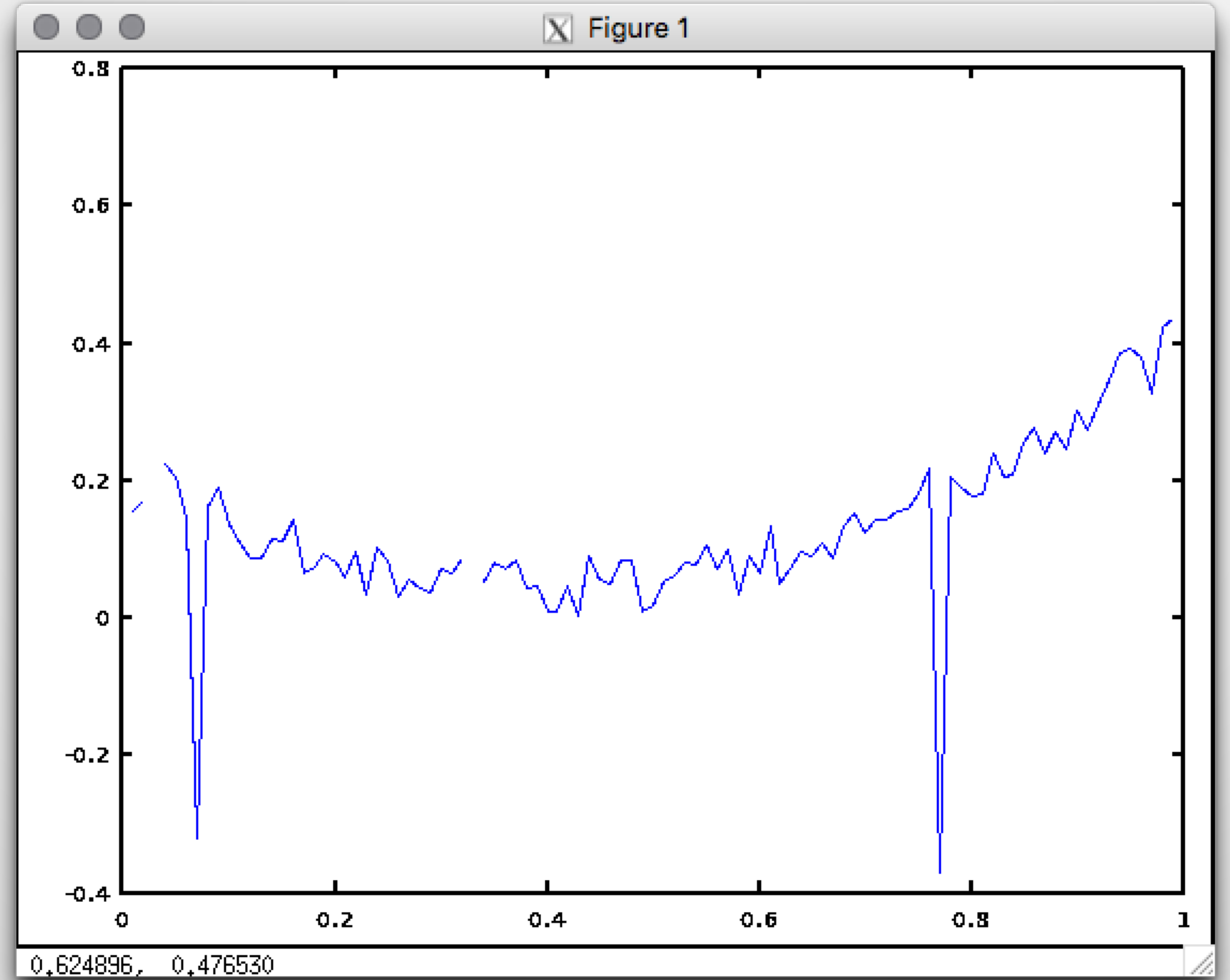
nb_res = 99;

p = Array{Float64}(undef, nb_res)
r = Array{Float64}(undef, nb_res)

for i in 1:nb_res
    ini = read(Inifile(), "res-$i.txt");
    p[i] = parse(Float64, get(ini, "main","parameter"))
    r[i] = parse(Float64, get(ini, "main","result"))
end

r[findall(abs.(r[1:end-1] - r[2:end]).>.1)] .= NaN
r[findall(isnan.(r))] .= Inf
#plot(r)
show(findmin(r))
~
~
~
~
~
~
N… > r.jl                    < jul… < 5% ≡ 1/18 ㏑ : 11
```

Second challenge



3.

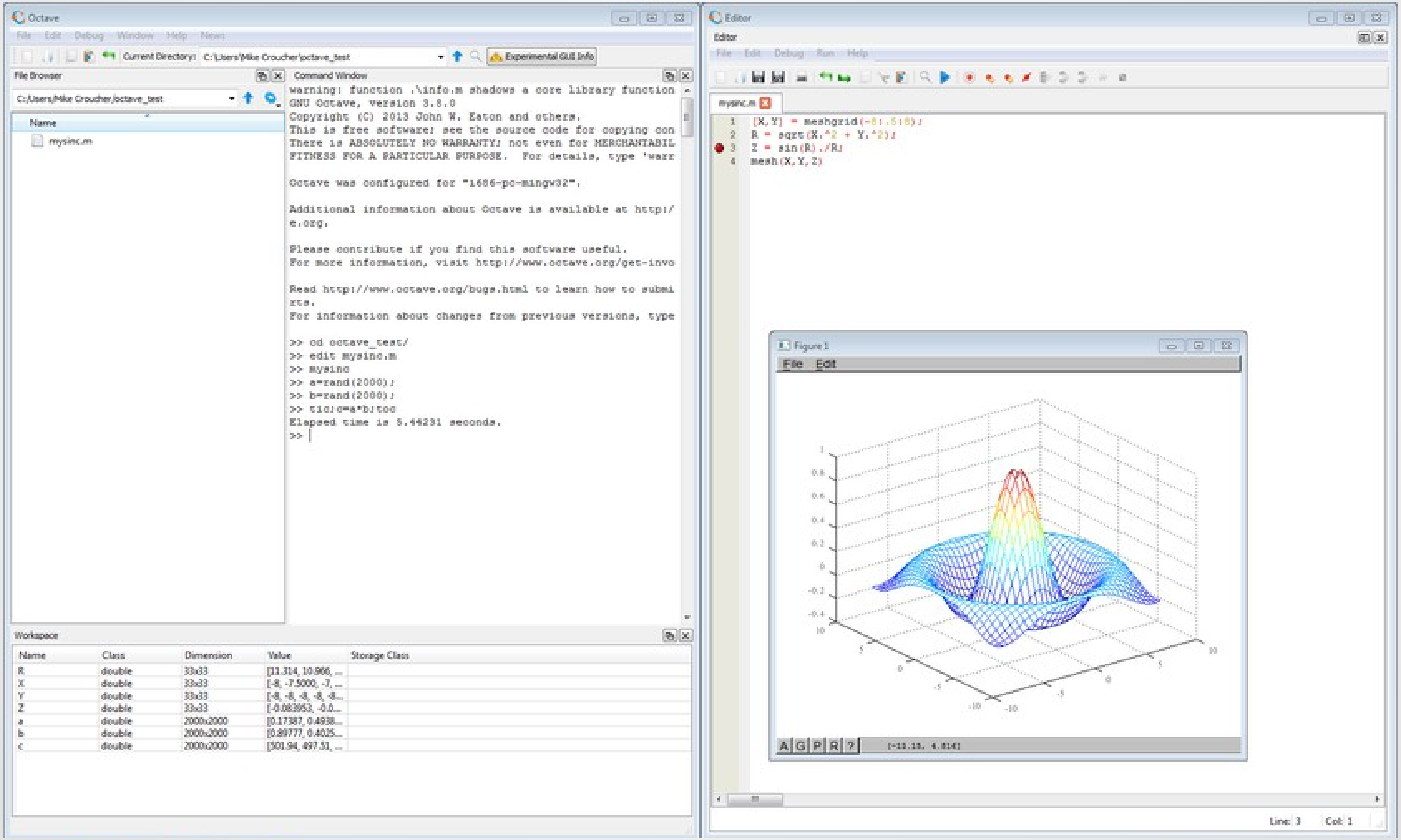
Graphical User Interfaces

Editing, debugging, accessing the doc, made easy

Literate programming

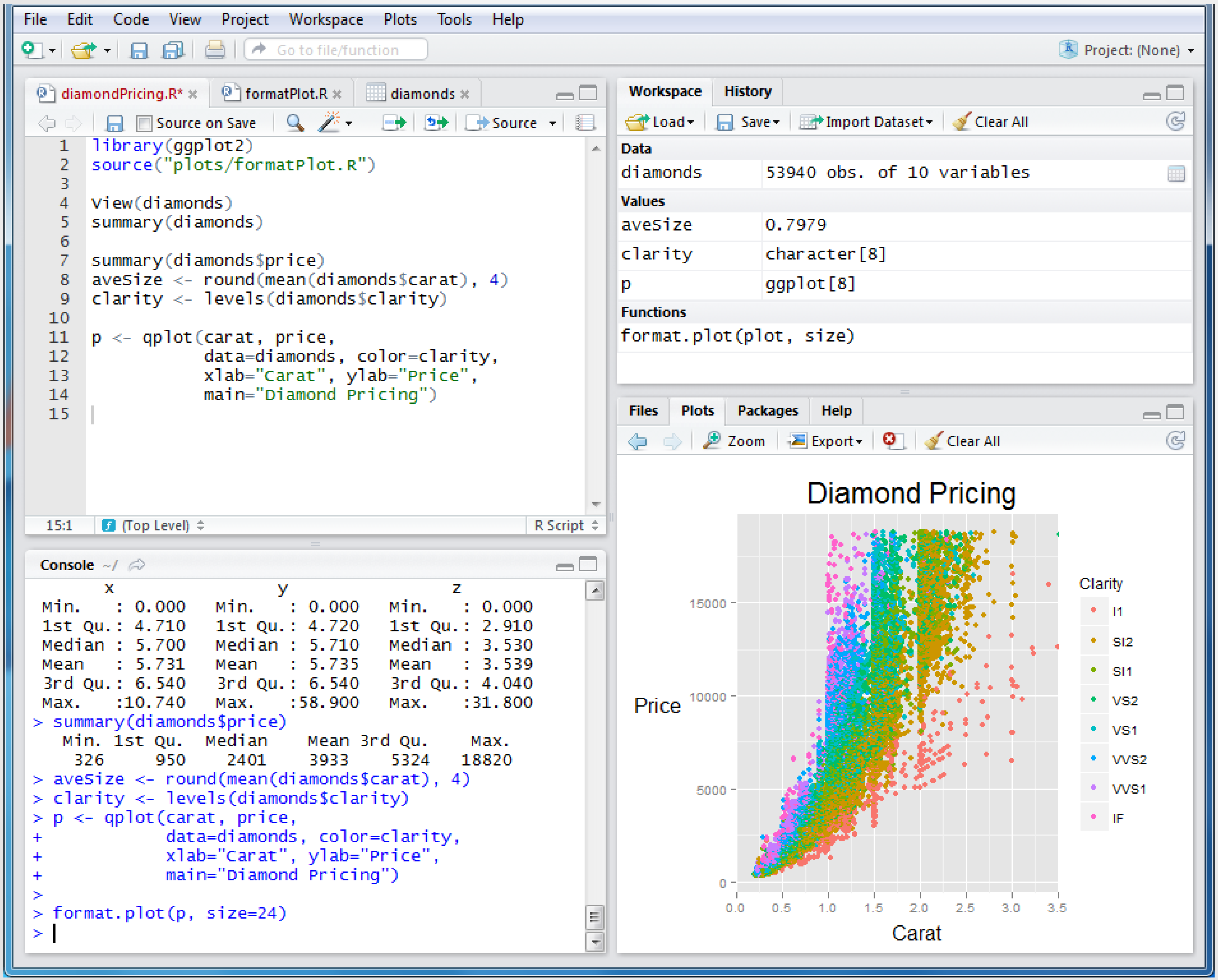
Authoring dynamic documents with code in them

Octave

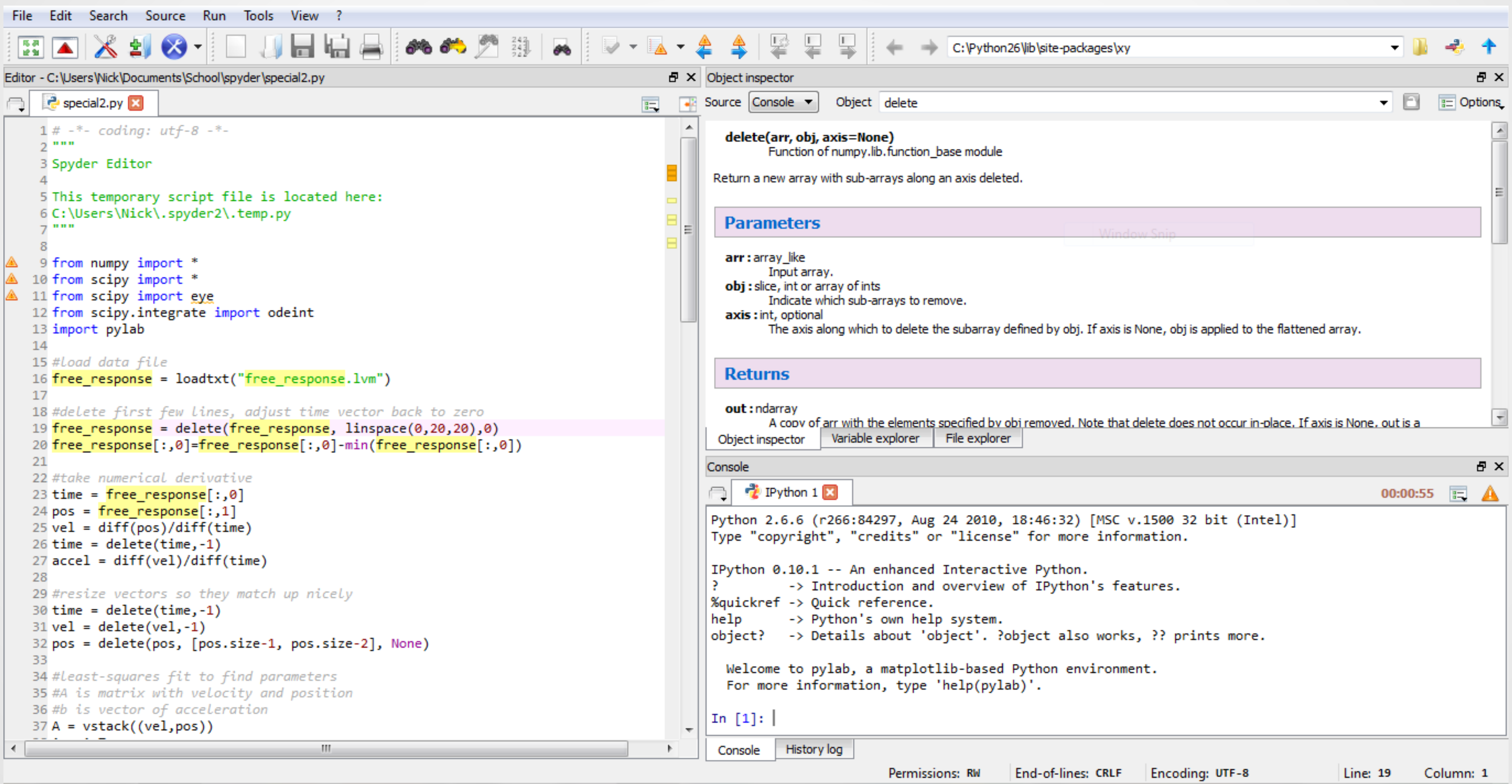


Octave
File Edit Debug Window Help News
Current Directory: C:\Users\Mike Croucher\octave_test
Experimental GUI Info
File Browser
C:/Users/Mike Croucher/octave_test
Name
mysinc.m
Command Window
warning: function .\info.m shadows a core library function
GNU Octave, version 3.8.0
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying con
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABIL
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warr
Octave was configured for "i686-pc-mingw32".
Additional information about Octave is available at http:/
e.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-invo
Read http://www.octave.org/bugs.html to learn how to submi
rts.
For information about changes from previous versions, type
>> cd octave_test/
>> edit mysinc.m
>> mysinc
>> a=rand(2000);
>> b=rand(2000);
>> tic;c=a*b;toc
Elapsed time is 5.44231 seconds.
>>
Workspace
Name Class Dimension Value Storage Class
R double 33x33 [11.314, 10.966, ...
X double 33x33 [-8, -7.5000, -7, ...
Y double 33x33 [-8, -8, -8, -8...
Z double 33x33 [-0.083953, -0.0...
a double 2000x2000 [0.17387, 0.4938...
b double 2000x2000 [0.89777, 0.4025...
c double 2000x2000 [501.94, 497.51, ...
Editor
File Edit Debug Run Help
mysinc.m
[X,Y] = meshgrid(-8:.5:8);
R = sqrt(X.^2 + Y.^2);
Z = sin(R)./R;
mesh(X,Y,Z)
Figure 1
File Edit
Line: 3 Col: 1

Rstudio


File Edit Code View Project Workspace Plots Tools Help
Go to file/function
Project: (None)
diamondPricing.R* formatPlot.R diamonds
Source on Save
Source
library(ggplot2)
source("plots/formatPlot.R")
View(diamonds)
summary(diamonds)
summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)
p <- qplot(carat, price,
data=diamonds, color=clarity,
xlab="Carat", ylab="Price",
main="Diamond Pricing")
15:1
(Top Level)
R Script
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
>
> format.plot(p, size=24)
>
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
15000
10000
5000
0
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

Spyder


File Edit Search Source Run Tools View ?
C:\Python26\lib\site-packages\xy
Editor - C:\Users\Nick\Documents\School\spyder\special2.py
special2.py
-*- coding: utf-8 -*-
"""
Spyder Editor

This temporary script file is located here:
C:\Users\Nick\.spyder2\.temp.py
"""

from numpy import *
from scipy import *
from scipy import eye
from scipy.integrate import odeint
import pylab

#load data file
free_response = loadtxt("free_response.lvm")

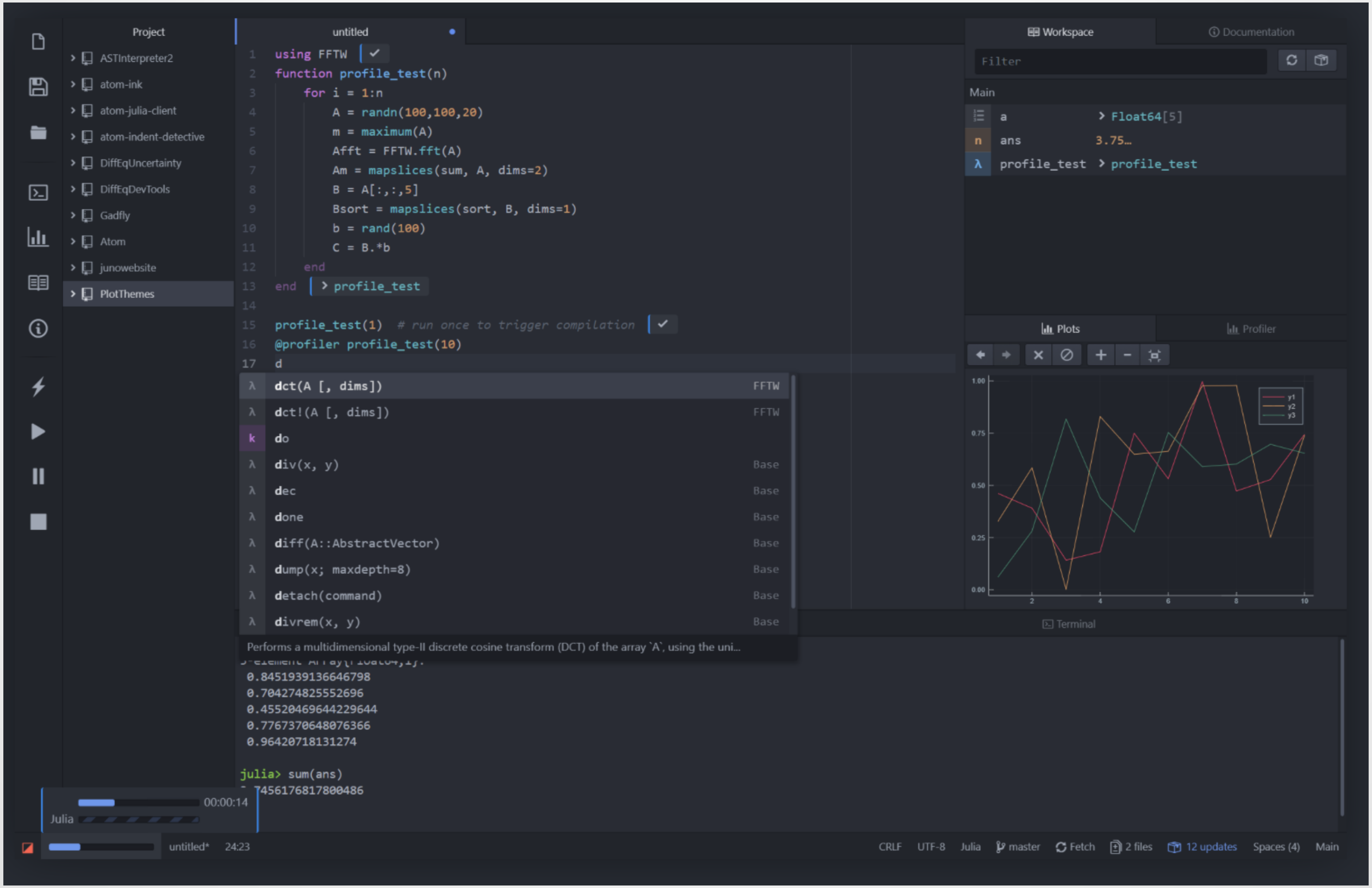
#delete first few lines, adjust time vector back to zero
free_response = delete(free_response, linspace(0,20,20),0)
free_response[:,0]=free_response[:,0]-min(free_response[:,0])

#take numerical derivative
time = free_response[:,0]
pos = free_response[:,1]
vel = diff(pos)/diff(time)
time = delete(time,-1)
accel = diff(vel)/diff(time)

#resize vectors so they match up nicely
time = delete(time,-1)
vel = delete(vel,-1)
pos = delete(pos, [pos.size-1, pos.size-2], None)

#least-squares fit to find parameters
#A is matrix with velocity and position
#b is vector of acceleration
A = vstack((vel,pos))
Object inspector
Source Console Object delete Options
delete(arr, obj, axis=None)
Function of numpy.lib.function_base module
Return a new array with sub-arrays along an axis deleted.
Parameters
arr : array_like
Input array.
obj : slice, int or array of ints
Indicate which sub-arrays to remove.
axis : int, optional
The axis along which to delete the subarray defined by obj. If axis is None, obj is applied to the flattened array.
Returns
out : ndarray
Object inspector Variable explorer File explorer
Console
IPython 1
00:00:55
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.
IPython 0.10.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.
Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.
In [1]:
Console History log
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 19 Column: 1

Juno


Project
ASTInterpreter2
atom-ink
atom-julia-client
atom-indent-detective
DiffEqUncertainty
DiffEqDevTools
Gadfly
Atom
junowebsite
PlotThemes
untitled
using FFTW
function profile_test(n)
for i = 1:n
A = randn(100,100,20)
m = maximum(A)
Afft = FFTW.fft(A)
Am = mapslices(sum, A, dims=2)
B = A[:,:,5]
Bsort = mapslices(sort, B, dims=1)
b = rand(100)
C = B.*b
end
end
profile_test
profile_test(1) # run once to trigger compilation
@profiler profile_test(10)
d
dct(A [, dims]) FFTW
dct!(A [, dims]) FFTW
do
div(x, y) Base
dec Base
done Base
diff(A::AbstractVector) Base
dump(x; maxdepth=8) Base
detach(command) Base
divrem(x, y) Base
Performs a multidimensional type-II discrete cosine transform (DCT) of the array `A`, using the uni...
0.8451939136646798
0.704274825552696
0.45520469644229644
0.7767370648076366
0.96420718131274
julia> sum(ans)
Julia
00:00:14
Workspace
Documentation
Filter
Main
a
Float64[5]
ans
3.75…
profile_test
profile_test
Plots
Profiler
Terminal
untitled* 24:23
CRLF UTF-8 Julia master Fetch 2 files 12 updates Spaces (4) Main

3.

Graphical User Interfaces

Editing, debugging, accessing the doc, made easy

Literate programming

Authoring HTML or LaTeX documents
with code and results in them

RMarkdown and KnitR

chunks.Rmd

```
R Code Chunks
=====================================

With R Markdown, you can insert R code chunks including plots:

```{r qplot, fig.width=4, fig.height=3, message=FALSE}
# quick summary and plot
library(ggplot2)
summary(cars)
qplot(speed, dist, data=cars) +
      geom_smooth()
```
```

RStudio: Preview HTML

Preview: ~/chunks.html | Save As | Publish

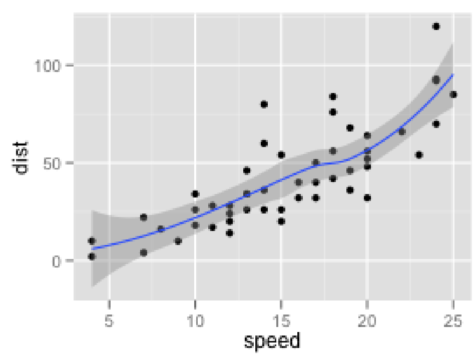
R Code Chunks

With R Markdown, you can insert R code chunks including plots:

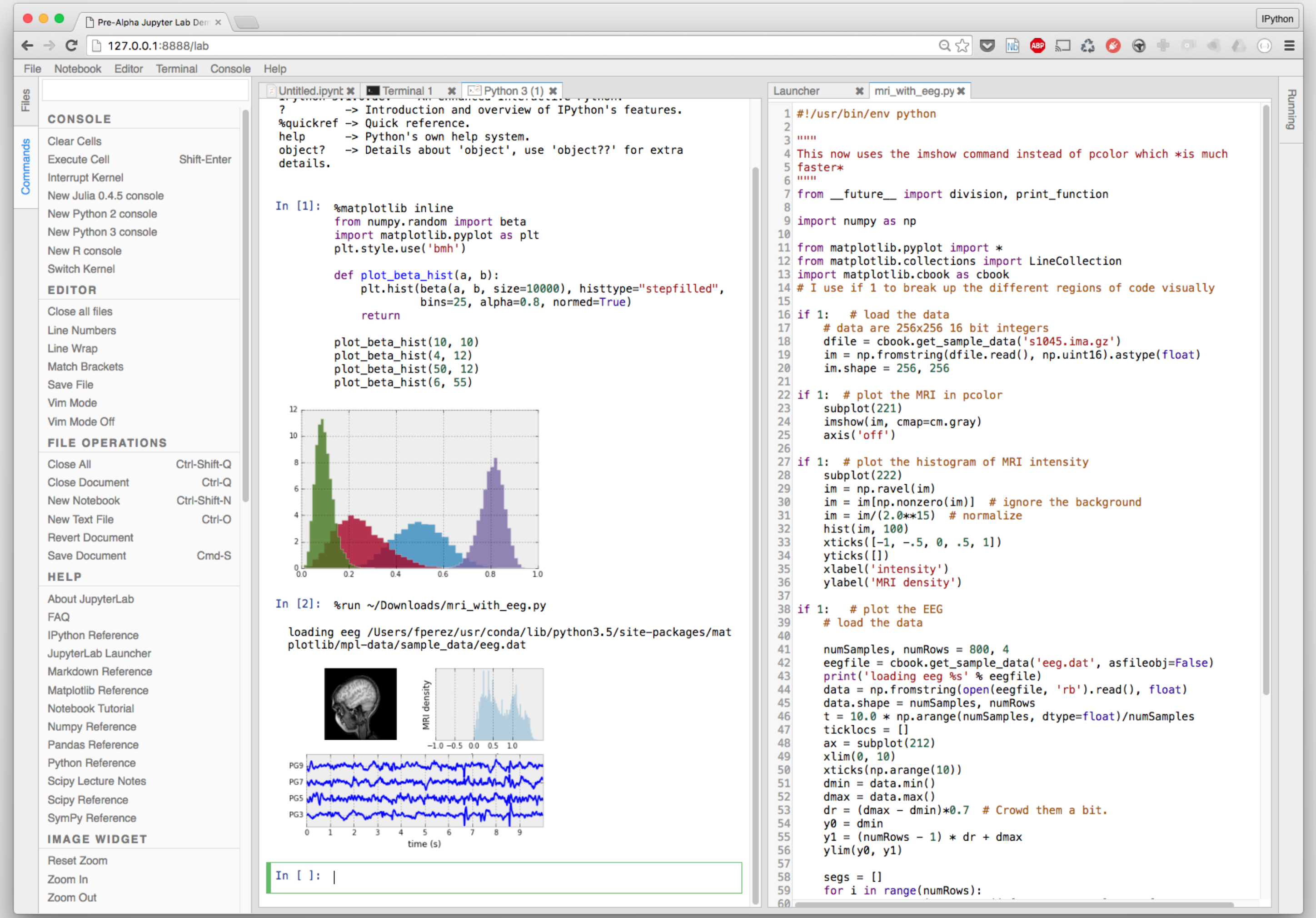
```
# quick summary and plot
library(ggplot2)
summary(cars)
```

```
##      speed           dist    
##  Min.   : 4.0   Min.   :  2  
##  1st Qu.:12.0   1st Qu.: 26  
##  Median :15.0   Median : 36  
##  Mean   :15.4   Mean   : 43  
##  3rd Qu.:19.0   3rd Qu.: 56  
##  Max.   :25.0   Max.   :120  
```

```
qplot(speed, dist, data = cars) + geom_smooth()
```



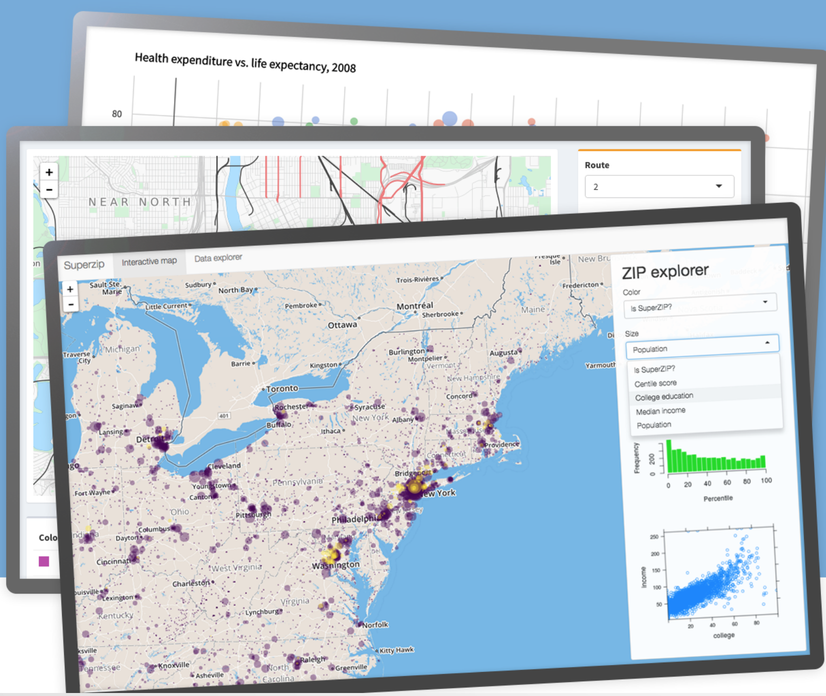
Jupyter notebooks



Shiny

Shiny from RStudio

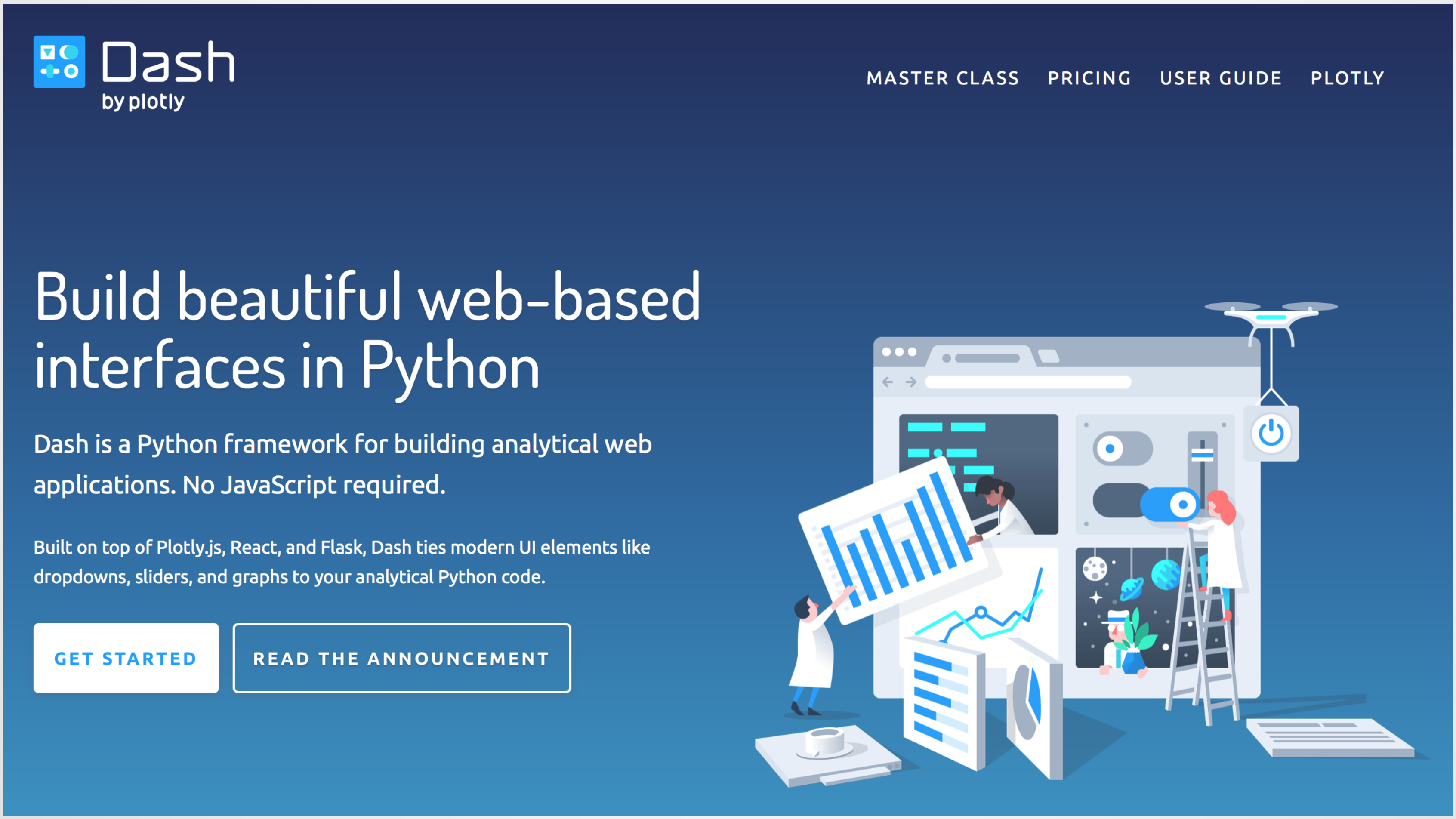
Get Started | Gallery | Articles | Reference | Deploy | Help | Contribute



Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

Dash

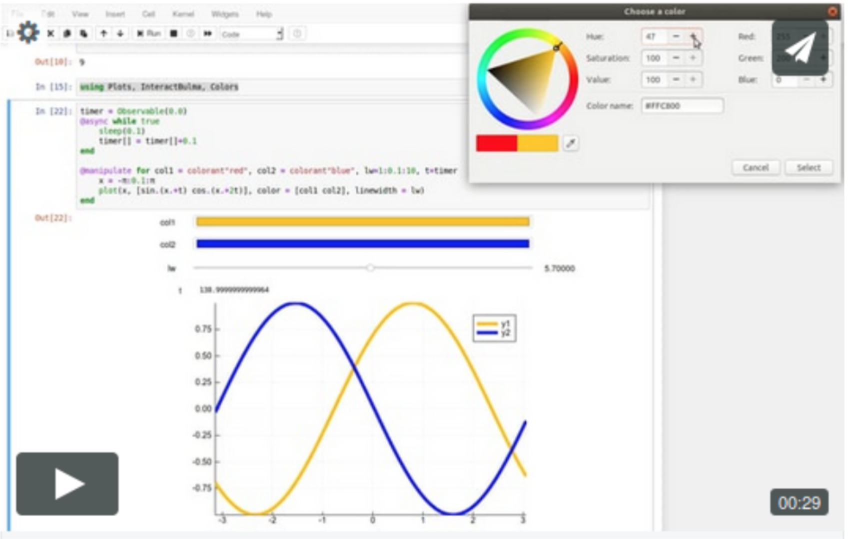

Dash
by plotly
MASTER CLASS
PRICING
USER GUIDE
PLOTLY
Build beautiful web-based interfaces in Python
Dash is a Python framework for building analytical web applications. No JavaScript required.
Built on top of Plotly.js, React, and Flask, Dash ties modern UI elements like dropdowns, sliders, and graphs to your analytical Python code.
GET STARTED
READ THE ANNOUNCEMENT

Interact.jl

Interact

build passing docs latest

Interact.jl allows you to use interactive widgets such as sliders, dropdowns and checkboxes to play with your Julia code:



Getting Started

To install Interact, run the following command in the Julia REPL:

```
Pkg.add("Interact")
```

4.

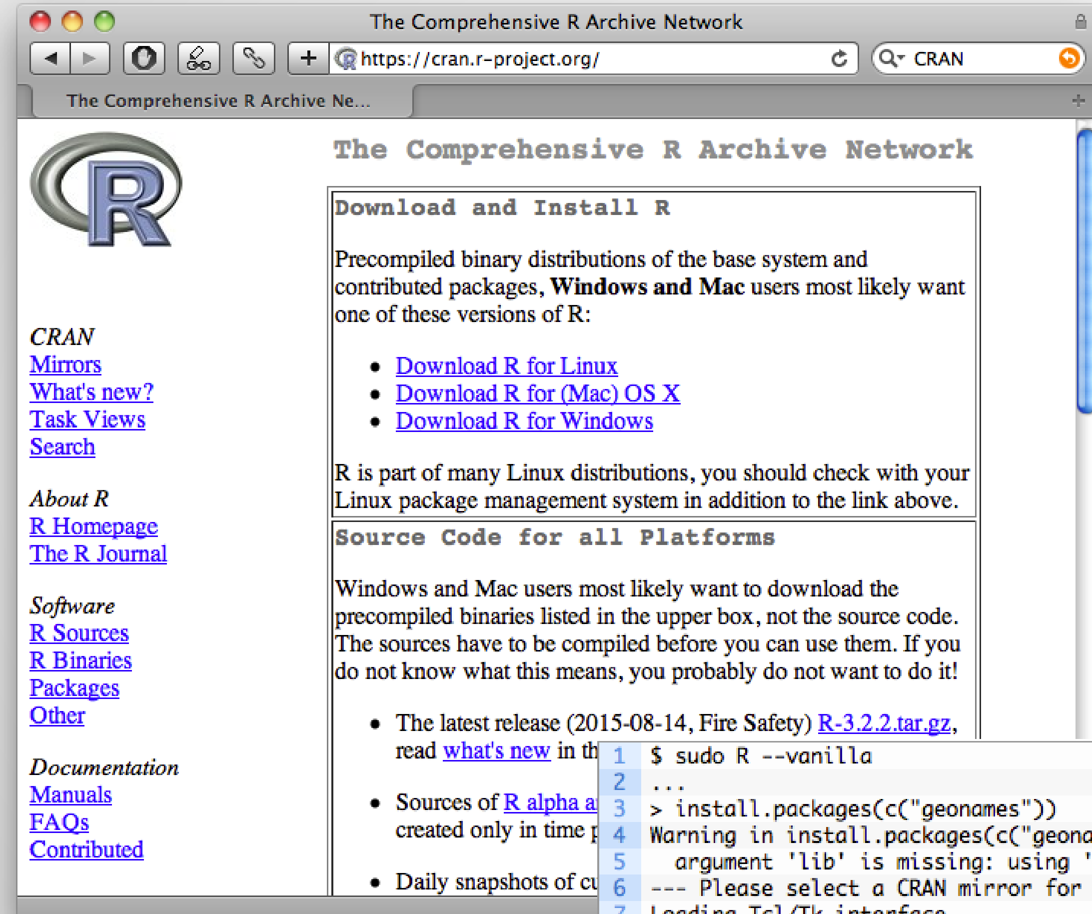
Extensions

Packages – Libraries – Modules

Octave Forge


```
>> pkg install -forge image
warning: creating installation directory C:\Octave\Octave-4.0.0\share\octave
warning: called from
 install at line 30 column 5
 pkg at line 405 column 9
For information about changes from previous versions of the image package, r
>> pkg list
Package Name  | Version | Installation directory
--------------+---------+-----------------------
        image |   2.4.0 | C:\Octave\Octave-4.0.0\share\octave\packages\image
```

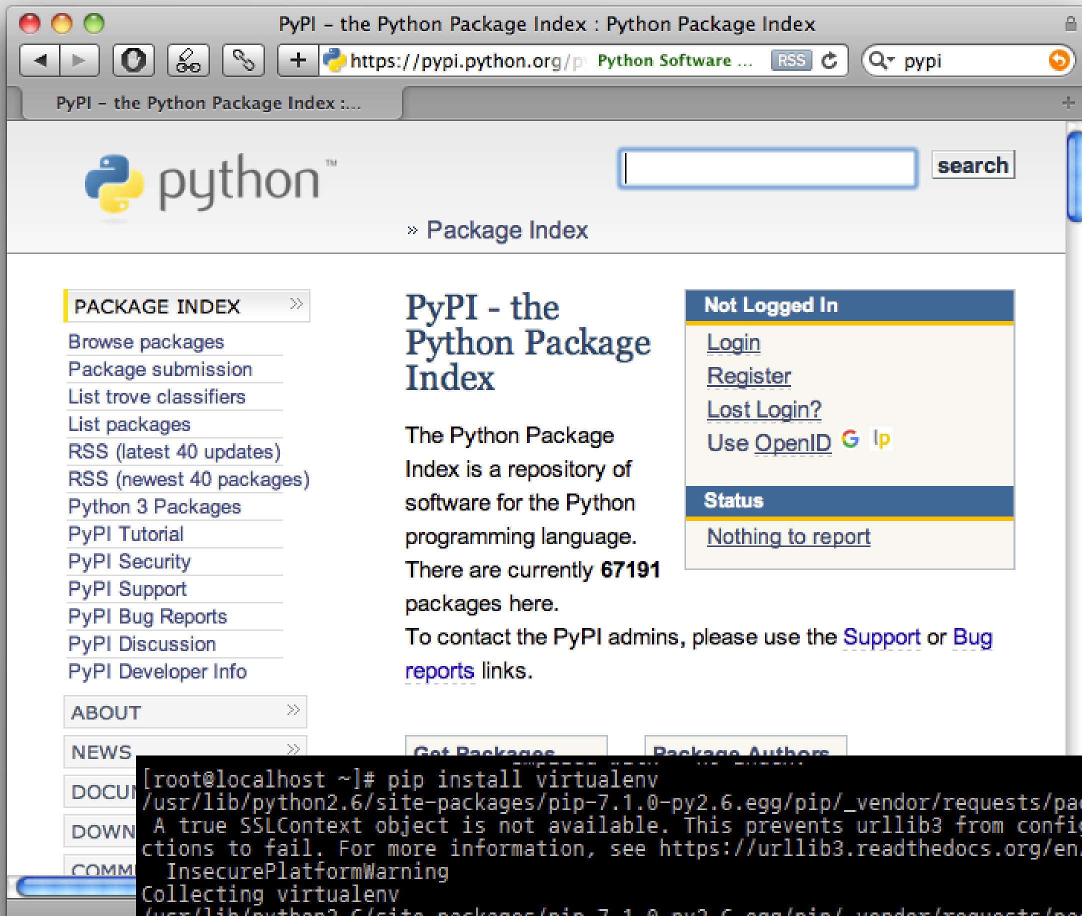
CRAN




```
$ sudo R --vanilla
...
> install.packages(c("geonames"))
Warning in install.packages(c("geonames")) :
  argument 'lib' is missing: using '/usr/local/lib/R/site-library'
--- Please select a CRAN mirror for use in this session ---
Loading Tcl/Tk interface ...
...
* DONE (geonames)

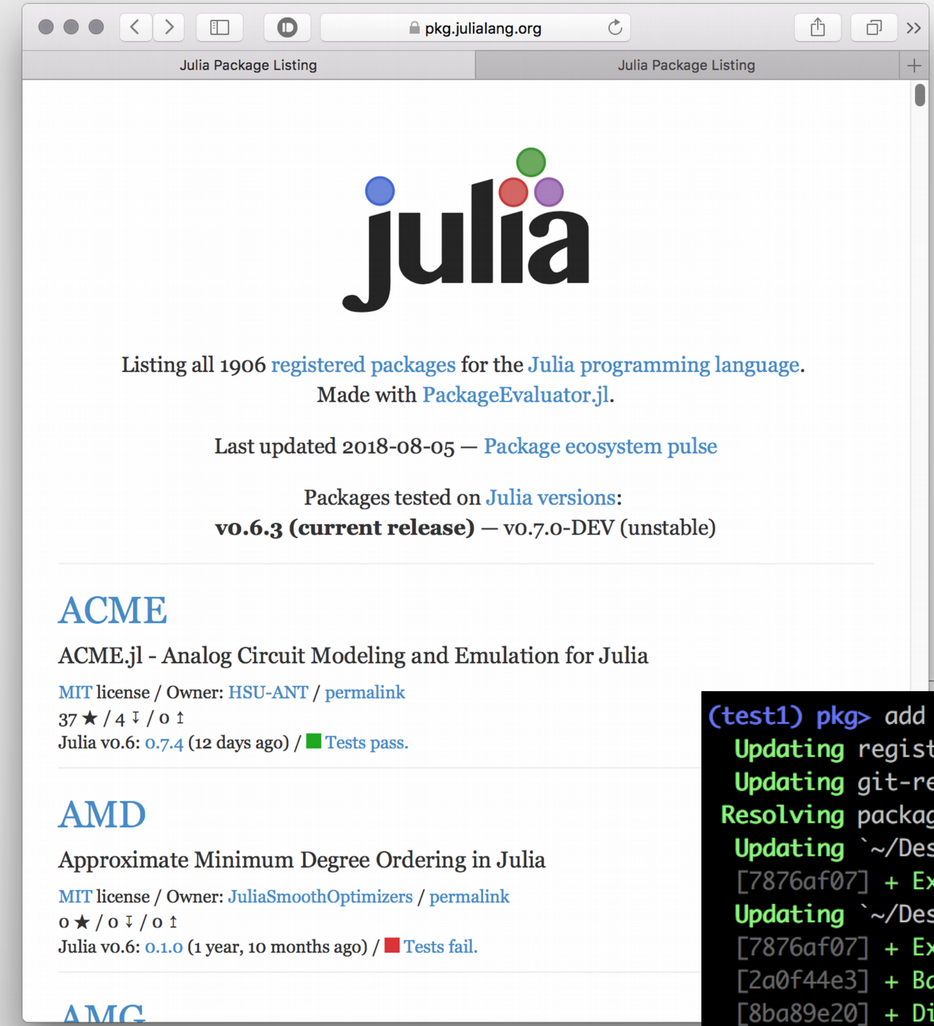
The downloaded packages are in
        /tmp/Rtmp3FziH3/downloaded_packages
```

PyPI




```
[root@localhost ~]# pip install virtualenv
/usr/lib/python2.6/site-packages/pip-7.1.0-py2.6.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:90: InsecurePlatformWarning:
 A true SSLContext object is not available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL conne
ctions to fail. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
Collecting virtualenv
/usr/lib/python2.6/site-packages/pip-7.1.0-py2.6.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:90: InsecurePlatformWarning:
 A true SSLContext object is not available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL conne
ctions to fail. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
  InsecurePlatformWarning
  Downloading virtualenv-13.1.0-py2.py3-none-any.whl (1.7MB)
    100% |████████████████████████████████| 1.7MB 201kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-13.1.0
[root@localhost ~]#
```

Julia package ecosystem




```
(test1) pkg> add Example
  Updating registry at `~/.julia/registries/General`
  Updating git-repo `https://github.com/JuliaRegistries/General.git`
 Resolving package versions...
  Updating `~/Desktop/hobby/julia/test/test1/Project.toml`
  [7876af07] + Example v0.5.1
  Updating `~/Desktop/hobby/julia/test/test1/Manifest.toml`
  [7876af07] + Example v0.5.1
  [2a0f44e3] + Base64
  [8ba89e20] + Distributed
  [b77e0a4c] + InteractiveUtils
  [8f399da3] + Libdl
```

5. General tips when it is slow

- Program thoughtfully:
  - Use vectorized functions
  - Avoid loops
  - Preallocate
  - Force type
  - Avoid copy-on-write
- Link to fast libraries (C/C++, Fortran, Java)
- Write low-level parts in C or Fortran
- Compile – jit
- Go parallel

6. Bridges

C.E.C.I

Python	→	R	http://rpython.r-forge.r-project.org/
Octave	→	Python	https://pypi.python.org/pypi/oct2py
R	→	Python	http://rpy.sourceforge.net/
Octave	→	R	https://cran.r-project.org/web/packages/RcppOctave
Python	→	Octave	https://github.com/daniel-e/pyoctave
R	→	Octave	http://www.omegahat.org/ROctave/
R	→	Julia	https://github.com/Non-Contradiction/JuliaCall
Julia	→	R	https://github.com/JuliaInterop/RCall.jl
Python	→	Julia	https://github.com/JuliaPy/pyjulia
Julia	→	Python	https://github.com/JuliaPy/PyCall.jl

Summary

Octave, R, Python (and Julia)

Much more programmer-friendly than C/C++/Fortran

Still able to use fast compiled code

Focus on the unsolved problems

Try all and choose one