Rewardingly Reproducible Research

September 21, 2016

1 Rewardingly Reproducible Research

1.1 Topics

- Python
- Git + Github
- Sumatra
- Campus HPC
- Mendeley

2 Using Python for everything

2.1 Replace Matlab with Python

```
In [1]: import numpy as np
        x = np.arange(0, 10, 1)
        print(x)
        I = np.eye(x.shape[0])
        print(I)
        y = I.dot(x)
        print(y)
```

```
[0 1 2 3 4 5 6 7 8 9]
[[ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  1.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]]
[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
```

2.2 Pandas is great for time-series or labeled data

http://nbviewer.jupyter.org/gist/wesm/4757075/PandasTour.ipynb

2.3 Write lower level code for speed

2.3.1 Numba (JIT)

```
In [3]: from numba import jit
        from numpy import arange
```

```python
# jit decorator tells Numba to compile this function.
# The argument types will be inferred by Numba when function is called.
@jit
def sum2d(arr):
    M, N = arr.shape
    result = 0.0
    for i in range(M):
        for j in range(N):
            result += arr[i,j]
    return result

a = arange(9).reshape(3,3)
print(sum2d(a))
```

```
36.0
```

2.3.2 Other Options

- Cython (precompiled C)
- f2py (Fortran)
- mpi4py

2.4 Using GPUs

- Either CUDA or OpenCL

```python
In [ ]: import pycuda.autoinit
        import pycuda.driver as drv
        import numpy

        from pycuda.compiler import SourceModule
        mod = SourceModule("""
        __global__ void multiply_them(float *dest, float *a, float *b)
        {
          const int i = threadIdx.x;
          dest[i] = a[i] * b[i];
        }
        """)

        multiply_them = mod.get_function("multiply_them")

        a = numpy.random.randn(400).astype(numpy.float32)
        b = numpy.random.randn(400).astype(numpy.float32)

        dest = numpy.zeros_like(a)
        multiply_them(
                drv.Out(dest), drv.In(a), drv.In(b),
                block=(400,1,1), grid=(1,1))
```

```python
In [ ]: import skcuda.linalg as culinalg
        import pycuda.gpuarray as gpuarray

        # put a and b onto the GPU
        a_gpu = gpuarray.to_gpu(a)
```

```
b_gpu = gpuarray.to_gpu(b)
c = culinalg.dot(a_gpu.T, b_gpu)
c.get() # get the results back from GPU memory
```

2.4.1 Example: OI

$$\hat{x} = x_b + W(y - Hx_b)$$
$$W = PH^T(R + HPH^T)^{-1}$$

x_b has dimension 6150 so P has dimension 6150x6150
Simple GPU code is 5x faster than highly optimized, multicore CPU code

2.5 Jupyter Notebook for interactive exploration

http://localhost:8888/notebooks/Example%20Notebook.ipynb

2.6 Useful python packages

- seaborn for plotting
- panda
- numpy
- scipy
- pycuda/pyopencl
- scikit-cuda linear alg on gpu (need cula)
- scikit-learn machine learning
- xarray NetCDF files
- pytables store stuff in HDF5 files
- fenics finite element
- theano deep learning
- mpi4py parallel computing
- numba just in time compiled code
- cython C like python code
- f2py call Fortran from python

2.7 Use Conda to install Python packages

- https://www.continuum.io/downloads
- Installs precompiled python packages so you don't need to find and install libraries (e.g. ATLAS for numpy)
- Can make separate environments for each project to keep package versions separate

2.8 Python is also a great general purpose language

3 Version Control

3.1 Git

- A distributed version control software
- Use branches freely, commit often
- No more code.v1, code.v2, etc.
- No need to worry that changes will break something
- Easy to see what changes have been made between code versions
- Always run simulations with version controlled code and keep track of that version in the output
- https://git-scm.com/doc

3.2 GitHub

- Keep copies of your git repos in the cloud
- Easily share code with others
- Free private repos (and other benefits) for students https://education.github.com/ or academic organizations
- https://guides.github.com/

4 Simulation Reproducibility: Sumatra

http://localhost:5006/satelliteOI/64880a9e5d87/

5 Campus HPC

- Free access with PI sponsor
- Mostly optimized to run code designed for the HPC environment (MPI etc)
- Can still run non-optimized code on a single node (12 to 28 cores, lots of memory) instead of on your personal computer
- El Gato has GPU nodes and Intel Phi nodes for code optimized for those (and seems to not keep track of compute time)
- http://rc.arizona.edu
- https://confluence.arizona.edu/display/UAHPC/Compute+Resources

5.0.1 Avoid typing your password everytime on the HPC

First, check if you have an SSH private key:

`ls ~/.ssh/id_rsa.pub`

(this is most common but not exhaustive)

If not, generate a key:

`ssh-keygen`

Finally copy the key to whatever logon host:

`ssh-copy-id user@elgato-login.hpc.arizona.edu`

Alternatively, copy the contents of `~/.ssh/id_rsa.pub` from your local machine to `~/.ssh/authorized_keys` on the remote machine.

6 Mendeley

Organize your reference library and easily add new references