INTRODUCTION (INDEX.HTML#CONTENT) RUNNING SOS (RUNNING.HTML#CONTENT)
SOS NOTEBOOK (NOTEBOOK.HTML#CONTENT) SOS WORKFLOW SYSTEM (WORKFLOW.HTML#CONTENT)
(https://github.com/vatlab) (https://vatlab.github.io/blog)
(https://twitter.com/ScriptOfScripts) (https://gitter.im/vatlab/SoS)
(https://www.youtube.com/channel/UCPsumGZ-C2NddWO54CvEdSw/videos)
(https://github.com/binder-examples/jupyter-sos)
(http://128.135.144.117:8000/)

SoS Workflow System

Notebook environment for both interactive data analysis and batch data processing

SoS Workflow System (CLI)

SoS supports Linux, Mac OSX, and Windows systems and requires Python 3 (https://www.python.org/) (version 3.6 or later) so you will need to install Python 3 if you do not have it installed locally. We recommend anaconda Python (https://www.continuum.io/downloads) because it is a complete

Python environment with many packages for scientific computing.

If you only need to use the command line interface of the SoS workflow system (e.g. to execute workflows on a remote server), you can install it with command

```
% pip install sos
```

or

```
% conda install -c conda-forge sos
```

SoS uses a subcommand system with subcommands such as `run` , `convert` , and `status` . You can get a list of subcommands using command

```
% sos -h
```

and usage of a particular subcomand using commands such as

```
% sos run -h
```

You can execute a SoS script `myscript` (or `myscript.sos`) in batch mode using command

```
% sos run myscript [options]
```

directly using command

```
% myscript [options]
```

if the script has shebang line

```
#!/usr/bin/env sos-runner
```

Please refer to chapter Command Line Interface (doc/documentation/User_Interface.html) of the SoS documentation for more details.

SoS Notebook

SoS Notebook can be used as both a polyglot notebook and an IDE for the SoS workflow system. It should be installed locally, or on a server to which you access remotely with a browser.

Live SoS server

If you are curious on what SoS and SoS Notebook are, you can try it out by clicking this link (http://128.135.144.117:8000/). This will lead you to our live SoS server from which you can click `New -> SoS` and create a SoS Notebook with all supported languages except for two proprietary ones (`MATLAB and SAS`).

SoS Docker Images

If you are using docker, you can run SoS directly using command

```
% docker run -it mdabioinfo/sos:latest /bin/bash
```

to enter a command prompt with sos command. More usefully, you can start a Jupyter server with R (https://www.r-project.org/) and IRkernel (https://github.com/IRkernel/IRkernel), Julia, Python, and SoS kernels, and many Python and R modules for data sciencists using command

```
% docker run -d -p 8888:8888 mdabioinfo/sos-notebook
```

After the docker is running in the background, you will need to find the name of the instance from the last column of the output of command `docker ps` and get the log message of the docker instance with `docker logs` followed by the name of your docker instance, e.g.

```
% docker logs eager_volhard
Execute the command: jupyter notebook
...
    Copy/paste this URL into your browser when you connect for t
he first time,
    to login with a token:
        http://localhost:8888/?token=754a646651c82657725be887a1a
2579ab69a702ba80ae4b3
```

You can then enter the URL in the log message to a browser and start working with a complete SoS environment. You could also set up the docker image to disable password as discussed here (https://github.com/jupyter/docker-stacks/tree/master/scipy-notebook) but using a Jupyter server without password is strongly discouraged.

You can even use this docker image for your daily data analysis if you make your local directory available to the Jupyter server using command

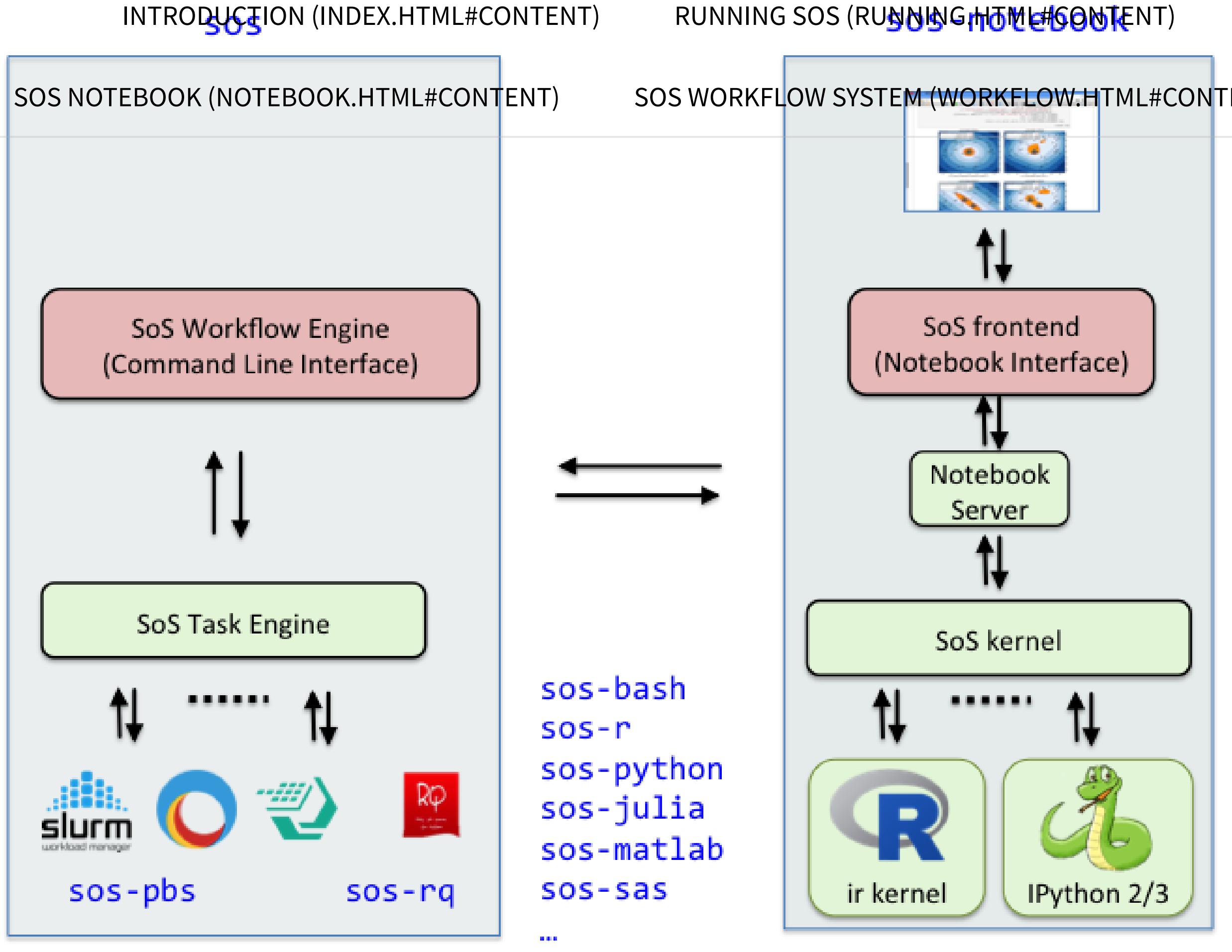
```
% docker run -d -p 8888:8888 -v $HOME:/home/jovyan/work mdabioi
nfo/sos-notebook
```

This command mounts your home directory (`$HOME`) to directory `work` under the home directory of the docker machine but you can mount any local directory to the docker image. This container is hosted at our public Jupyter server (http://ec2-34-192-184-206.compute-1.amazonaws.com:8000/) from which you can open our sample notebooks and create your own notebooks without installing anything.

Note: If you get an error message stating `Bind for 0.0.0.0:8888 failed: port is already allocated`, your local port `8888` is already taken by some other processes and you can use options such as `-p 9999:8888` to use another local port. Using a different port actually allows you to execute multiple instances of the docker image.

Local installation

SoS consists of two major parts, the **SoS Workflow Engine** and **SoS Polyglot Notebook**, each with a number of extension modules:



SoS Workflow Engine

- **sos** (https://pypi.python.org/pypi/sos/): SoS workflow engine with its command line interface
- **sos-pbs** (https://pypi.python.org/pypi/sos-pbs/): PBS task engine for Torch, Slurm, IBM LSF etc
- **sos-rq** (https://pypi.python.org/pypi/sos-rq/): rq (http://python-rq.org/) task engine for SoS
- **sos-bioinfo** (https://pypi.python.org/pypi/sos-bioinfo/): extension for bioinformatic applications

SoS Notebook

- **sos-notebook** (https://pypi.python.org/pypi/sos-notebook/): Core sos-notebook module
- **sos-notebook** (https://pypi.python.org/pypi/sos-bash/): SoS extension for shell scripts
- **sos-javascript** (https://pypi.python.org/pypi/sos-javascript): SoS extension for `JavaScript and Node.js`
- **sos-julia** (https://pypi.python.org/pypi/sos-julia/): SoS extension for `Julia`
- **sos-matlab** (https://pypi.python.org/pypi/sos-matlab/): SoS extension for `MATLAB and Octave`

- **`sos-python`** (https://pypi.python.org/pypi/sos-python/): SoS extension for `Python2` and `Python3`
- **`sos-r`** (https://pypi.python.org/pypi/sos-r/): SoS extension for R
- **`sos-sas`** (https://pypi.python.org/pypi/sos-sas/): SoS extension for SAS
- **`sos-stata`** (https://pypi.python.org/pypi/sos-stata/): SoS extension for Stata

After making sure that you have Python 3.6+ installed, you can install `sos`, `sos-notebook` and extension modules with commands such as

```
% pip install sos
% pip install sos-notebook
% pip install sos-r
```

After the installation of `sos-notebook`, you will need to register the sos kernel to Jupyter using command

```
% python -m sos_notebook.install
```

After verifying the `sos` and the kernels you would like to use are in the output of

```
% jupyter kernelspec list
```

you can start a Jupyter server with commnad

```
% jupyter notebook
```

and choose `SoS` as the kernel for a new notebook. Please refer to Notebook Interface (doc/documentation/Notebook_Interface.html) of the SoS documentation for details.

Certain features of SoS Notebook require optional python modules. They are not required but recommended for a complete SoS environment. These modules include

- graphviz (https://pypi.org/project/graphviz/), imageio (https://pypi.org/project/imageio/), and pillow (https://pypi.org/project/Pillow/) or PIL (https://pypi.org/project/PIL/) for the preview of `dot` files generated by the `-d` option of `sos run`.
- pysam (https://pypi.org/project/pysam/) for the preview of bam and sam files.

Finally, if you are using `vim`, you can use command

```
% python -m sos.install
```

to install a vim extension of sos to enable syntax highlighting when editing `.sos` files.

Jupyter Lab

JupyterLab is still developing but is ready for users, so is the JupyterLab extension of SoS called jupyterlab-sos (https://github.com/vatlab/jupyterlab-sos). If you are already using JupyterLab, you can install the jupyterlab extension of SoS using command

```
% jupyter labextension install jupyterlab-sos
```

All features of SoS Notebook are expected to work for JupyterLab, except for those that rely on the side panel. For example, magics `%toc` , `%preview` etc now only send result to the main notebook, and clicking the task ID will not trigger `%taskinfo` . We are working with the JupyterLab team to resolve these issues and can hopefully get it done befor the final release of JupyterLab.

Supported Languages

To use a language in `SoS` , your system should have the corresponding interpreters (`bash` , `python` , `julia`) and Jupyter kernel (e.g. `ir for R` , `ijavascript for JavaScript`) installed. The installation process varies from language to language, and can be quite troublesome depending on language and operating system. In general, you should

1. Verify the availability of an interpreter by executing the interpreter from a command line (e.g. run `julia` to check if `julia` is installed.
2. Install the jupyter kernel for the interpreter, and verify if the corresponding kernel is installed correctly by running
   ```
   jupyter kernelspec list
   ```
3. Start Jupyter, create a notebook with the kernel and check if you can use the kernel correctly.
4. Visit the language-specific sections below to install any required modules that are used for data exchange between SoS and the kernels.

Bash

`bash` is generally available under Linux and MacOSX so you only need to install the Bash Kernel (https://github.com/takluyver/bash_kernel) for Jupyter. The `bash` command can be installed under windows but is generally not tested.

JavaScript

Although there appears to be several Jupyter Kernels, SoS is only tested with the iJavaScript kernel (https://github.com/n-riesco/ijavascript). To use this kernel, please follow iJavaScript kernel homepage (https://github.com/n-riesco/ijavascript) to install `iJavaScript kernel`.

Julia

Start `Julia` and install iJulia (https://github.com/JuliaLang/IJulia.jl) by following iJulia Kernel homepage (https://github.com/JuliaLang/IJulia.jl). After that, install feather.jl (https://github.com/JuliaStats/Feather.jl), DataFrames.jl (https://github.com/JuliaData/DataFrames.jl) and NamedArrays.jl (https://github.com/davidavdav/NamedArrays.jl) with commands

```
using Pkg
Pkg.add("Feather")
Pkg.add("DataFrames")
Pkg.add("NamedArrays")
```

Note that it is important to set

```
ENV["JUPYTER"] = "/path/to/jupyter"
```

in Julia before running

```
Pkg.add("IJulia")
```

so that `IJulia` can be installed to the existing installation of Jupyter.

Finally on the SoS side, the Python feather-format (https://github.com/wesm/feather) module should be installed, most likely with command

```
conda install -c conda-forge feather-format
```

to facilitate the exchange of data frames, and please do not forget to install the Julia language module

```
pip install sos-julia
```

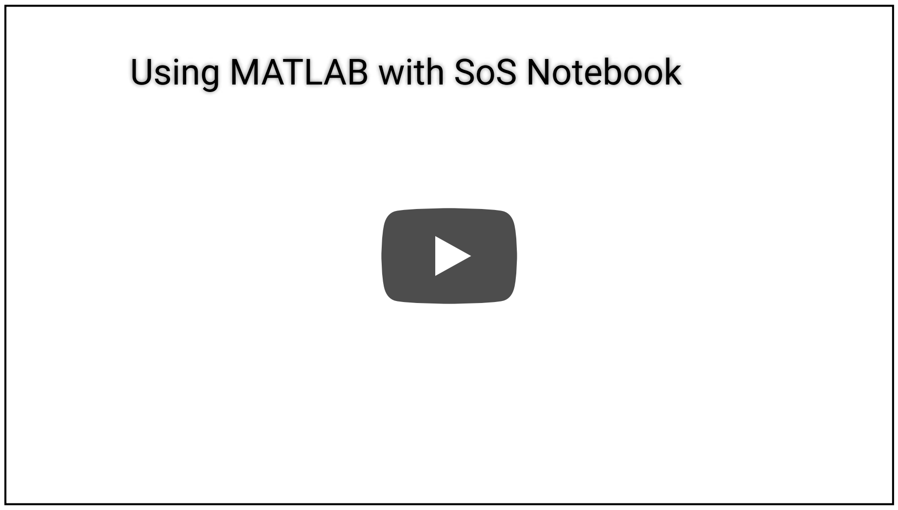
MATLAB

Because `SoS` requires `Python 3.6` and only `MATLAB` version 2017b+ supports this version of `Python`, you will need to have a working version of `MATLAB 2017b` or later installed on your system. Then, you will need to install matlab engine for Python (https://www.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html), which typically involves the execution of command `python setup.py install` under `matlabroot\extern\engines\python`.

Because of a bug with usage statistics collection in MATLAB 2017b (https://mathworksservicerequest.secure.force.com/apex/cp_case_detail?cc=us&id=5000Z00000tgPA0), you will need to turn off MATLAB's usage statistics collection system before you use `MATLAB 2017b` with `SoS`. To resolve this issue, You can opt-out of usage statistics collection by using the following steps: On the `Home` tab, in the `Environment` section, click `Preferences`, then select `MATLAB > General` in the `Preferences` window. Uncheck the box `Improve MATLAB by sending user experience information to MathWorks`.

There are two different implementations of `MATLAB` kernels for Jupyter `matlab_kernel` (https://github.com/Calysto/matlab_kernel) and imatlab (https://github.com/imatlab/imatlab). Because of a bug with `matlab_kernel` (https://github.com/vatlab/sos-matlab/issues/2), you should install the `imatlab` kernel by following instructions on the imatlab homepage (https://github.com/imatlab/imatlab).

Because `MATLAB` is among the most difficult languages to configure, we have recorded a video Using MATLAB with SoS Notebook (https://youtu.be/t9ohJZnuanc) with detailed instructions on how to configure SoS Notebook to work with MATLAB.



Octave

After installing octave, install the octave kernel (https://github.com/Calysto/octave_kernel) by following instructions on the octave kernel homepage (https://github.com/Calysto/octave_kernel).

For transferring Python's DataFrame and its equivalences in other languages, you will need to install the dataframe (https://octave.sourceforge.io/dataframe/index.html) package using the following command:

```
octave --eval 'pkg install -forge dataframe'
```

Python 2

If you still have Python 2.x installed on your system and would like to use it with `SoS`, you will need to

- Place executable `python2` or `python2.7` in your `$PATH` and use action `python2` for python2 scripts.
- Install python2 kernel following directions here (http://ipython.readthedocs.io/en/stable/install/kernel_install.html).

Python 3

Jupyter comes with working `Python3` kernel so no further installation is needed.

R

To use `R` with `SoS`, you will need to install the following components:

- IRKernel (https://github.com/IRkernel/IRkernel) kernel for Jupyter.
- R feather (https://cran.r-project.org/web/packages/feather/index.html) library.
- Python feather-format (https://github.com/wesm/feather) module, which can be installed with command

  ```
  conda install -c conda-forge feather-format
  ```

If you have a working R installation, you can install `irkernel` and `feather` in R with commands

```
install.packages('devtools')
devtools::install_github('IRkernel/IRkernel')
IRkernel::installspec()
install.packages('feather')
```

If you are using anaconda and do not have `R` installed, you can install `R` and required packages using commands

```
conda install -c r r-essentials r-feather
conda install -c conda-forge feather-format
```

although there have been several issues with certain versions of conda (e.g. (libgcc_s_seh-1.dll under windows (https://github.com/ContinuumIO/anaconda-issues/issues/777) and missing libreadline under mac (https://github.com/ContinuumIO/anaconda-issues/issues/6312)).

Ruby

To use Ruby with SoS, you will need to install iRuby (https://github.com/SciRuby/iruby) by following iRuby Kernel homepage (https://github.com/SciRuby/iruby). After that, install daru (https://github.com/SciRuby/daru) and NMatrix (https://github.com/SciRuby/nmatrix) with command:

```
% gem install daru nmatrix
```

SAS

With a local or remote `SAS` installation of version 9.4 or higher, you will need to install `sas-kernel` (https://github.com/sassoftware/sas_kernel) and configure it to connect it to your `SAS` installation.

Note:

1. You will need SAS version 9.4 or higher to use `sas_kernel` .
2. The SAS Unversity Edition runs a jupyter server inside a Virtual Machine without ssh access. Although you can use this version to learn SAS and Jupyter, it is not possible to use it with SoS.

Note for Windows Users:

1. Please follow the guideline for configuration (https://sassoftware.github.io/saspy/install.html#configuration). The integrated object method (IOM) connection method is the only option for Windows.
2. The paths of `classpath` in sascfg.py (sascfg_personal.py) might need to be updated manually.
3. In sascfg.py (sascfg_personal.py), please change `SAS_config_names` to a list only containing one option that you want to use. You cannot choose the `SAS_config_names` in `sos_notebook`. For example, if winlocal is your choice, then simply change the code to `SAS_config_names = ['winlocal']`.

Stata

You will need to install Stata (https://www.stata.com/) and then the stata_kernel (https://github.com/kylebarron/stata_kernel).

TypeScript

After installing `npm` (https://www.npmjs.com/), `typescript` (https://www.typescriptlang.org/), you should install the iTypeScript kernel (https://github.com/nearbydelta/itypescript).

Notes

Remote access

One of the advantages of using a Jupyter notebook is the ability to access the notebook remotely. For example, you can start a Jupyter server from your office computer and connect to it from you home (as long as there is no firewall that blocks the assigned port).

The jupyter documentation (http://jupyter-notebook.readthedocs.io/en/latest/public_server.html) provides detailed instructions on how to start a Jupyter notebook server that accepts external connection. Generally speaking, you should run command

```
>>> from notebook.auth import passwd
>>> passwd()
```

from a Python shell to get `sha` presentation of a password. Generate a new configuration file (`~/.jupyter/jupyter_notebook_config.py`) with command

```
jupyter notebook --generate-config
```

and modify it with lines such as

```
c.NotebookApp.ip = '*'
c.NotebookApp.password = u'sha1:...<your hashed password here>'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 8888
```

Then, after you start your notebook server using command

```
% jupyter notebook
```

You should be able to access it remotely with URL

```
http://url-or-ip-of-notebook-server:8888/
```

`virtualenv` or `pipenv`

If you are using virtualenv or pipenv, you might need to remove the `sos` kernel installed globally with command

```
% jupyter kernelspec remove sos
% python -m sos_notebook.install
```

to install sos for the particular `python` interpreter of the virtual env.

Windows

Windows systems lack native support for some of the tools that SoS uses and it is generally more difficult to set up Jupyter kernels for different languages. We therefore do not recommend the use of SoS and SoS Notebook under windows for novice users.

The best way to use SoS under windows is to use a Linux subsystem. You could enable Linux subsystem for windows (https://msdn.microsoft.com/en-us/commandline/wsl/about) if you have a Windows 10 system with Developer Mode enabled, or use one of the Linux subsystems such as Cygwin (https://www.cygwin.com/), MinGW (http://www.mingw.org/), or MSYS2 (http://www.msys2.org/). We generally recommend the use of MSYS2 because of its pacman package manage system.

To install MSYS2,

- Install MSYS2 from MSYS2 homepage (http://www.msys2.org/)
- Start MSYS2, run

  ```
  $ pacman -S openssh rsync git
  ```

- Add `c:\msys64\usr\bin` (adapt to your installation) to environment variable `$PATH` so that commands `rsync` , `rcp` , `ssh` ,and `git` are available to sos.

Note that

- This configuration allows executing tasks generated from a windows localhost on remote Linux and Mac OSX hosts (task queues). **Remote execution on a windows host is not yet supported**.
- Installation of `git` is optional especially if you already have git for windows (https://git-scm.com/downloads) installed (which is also based on msys2).
- You might want to install ConEmu (https://conemu.github.io/) as a (much better) replacement for Windows command console.
- You will need to set up `$HOME` properly to use ssh and public key authentication with other machines. This page (https://github.com/valtron/llvm-stuff/wiki/Set-up-Windows-dev-environment-with-MSYS2) provides a nice summary of the steps.

(https://www.mdanderson.org/)