Core | Research & Development

RLab : A Lightweight Platform-Independent Data Science Research Lab (Docker image : xtnc/rlab)

Motivation

The headache of setting up different research environments in python for data science is no stranger to any data scientist spanning from amateurs and enthusiasts to experts.
Managing different versions of Python, subsequent packages, dependencies' versions and virtual environments in Anaconda often implies a lot of overhead work in repetitive and time consuming installations, solving platform incompatibility, etc.
These make data science research very not suited for a modern AGILE process and hard for the users to recreate research results from their personal computer to another computer without going through the problems mentioned above.

RLab is created with the vision of setting up an easy to use platform-independent virtualized environment for data science research in Python with Jupyter Notebook.

RLab utilize the application-level virtualization features of Docker (containers) to provide a platform independent and consistent research lab in Jupyter.

Purpose

Easy accessible data science environments in Python for quantitative research (targeting mostly but not limited to options)
Custom Built-In Library classes for proprietary calculations and data manipulations
Portable and consistent dependencies in the environment.

Core Features (v1.0.1)

- Fully customizable template python (Python 3.5) file that will be pre-runned by the kernel for all existing ipynb files
- Extensive OptionLib Library wrapper for Options data pulling from Yahoo Finance
- Black-Scholes model computing for delta, gamma, theta, rho and kappa (European Options Calculations)
- Multi-threaded background process for automatic data updating and building in dataframes
- 3D Plotting for Implied Volatility, Deltas, Gammas, Kappa, Thetas and Rhos
- Accessibility to custom libraries and function
- Consistency in packages and dependencies versioning

DOCUMENTATION

Han Y. Xiao, Technical Partner | Xiao Theodore & Co.

User Guide

For any member who wish to utilize the Research Lab

1. From CLI using Docker (Linux, OSX, Windows):
    a. Have Docker client running on your local machine and/or host (ref. https://www.docker.com/community-edition#/download)
    b. Sign up for community free account with Docker Hub
    c. Once installed docker, login into docker CLI with your new docker hub credentials

```
$ docker login
```

    d. Then pull the image (Docker terminology for a grouping of executable files) of RLab

```
$ docker pull xtnc/rlab:latest
```

    e. Then simply run the docker image in a container on your host with the following command

```
$ docker run -i -t -p 8888:8888 -v <PATH TO RESEARCH>:/researchs xtnc/rlab
```

    Don't forget to replace <PATH TO RESEARCH> with the absolute path to your local research repository

    f. Then simply follow the returned URL in your browser "et voila", you have an instance of Jupyter Notebook running

2. From XTNC Repository (BitBucket) [For Associates and Partners of Xiao Theodore & Co.]
    a. Make sure that you have set up/obtained a pair of RSA Keys for your account on BitBucket for XTNC repositories.
    Note: You must be an associate or a partner to obtain access
    b. Navigate into a research repository located in Research & Development Bitbucket project (e.g. Options_Research) and pull the repository using SSH and your keys

```
$ git pull git@bitbucket.org:xtnc_core/options_research.git
```

    c. Make sure that you have docker installed on your host machine and that you are logged in, (if not, refer to earlier instructions)
    d. Navigate into Options_Research directory and simply run the shell scripts start.sh (You might need to make it executable first)

```
$ cd Options_Research
```

```
$ sudo chmod 755 start.sh
```

```
$ ./start.sh
```

e. Et Voila! Follow the url and token (copy paste the *http://0.0.0.0:8888/?token=2948d (…)d351443dca)* from the command line into your browser and start your research! When you are done, save your ipynb file and a simple [CTRL] + C will terminate the Lab
f. Then everytime you need to restart your Lab and research, you just need to run

```
$ ./start.sh
```

Developer Guide

1. Pull the research lab repository after configuring your SSH keys

```
$ git pull git@bitbucket.org:xtnc_core/research-lab.git
```

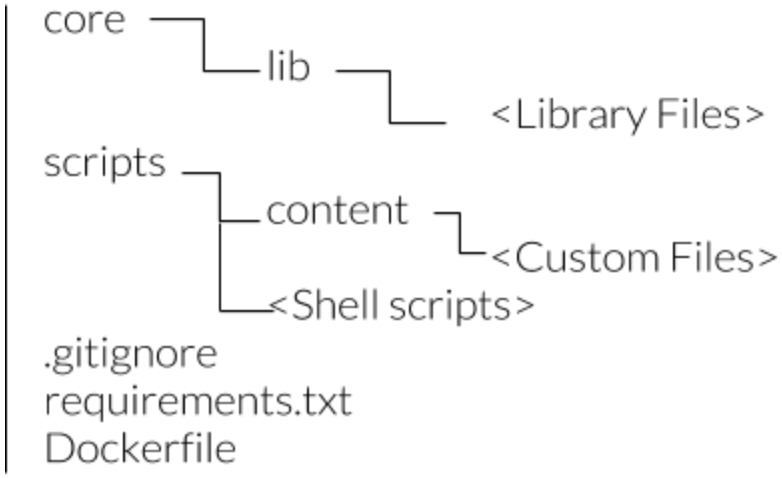
Then navigate into the directory

```
$ cd Research_Lab
$ git checkout feature/xyz
```

Then navigate into the directory

```
$ ls -l
```

The structure should look something like this



If you want to add Library classes you should put them into core.lib
When you want to add header ipynb template you can edit startup.ipy in scripts.content
To change the style you can edit custom.css in scripts.content
It is not recommended to change the rest although feel free to do so, your pull request will be reviewed anyway.
Note: Do not forget to add newly added python dependencies' names in the requirements.txt

2. Building the image after modifications and pull requests

After setting up docker, logging in, navigate to the root of the repository and build the new image

```
$ docker build -t <username>/rlab .
```

Alternatively you can add a version tag like this

```
$ docker build -t <username>/rlab:<tag>
```

Then you can push it to your own Docker hub repository like this
Double check that the image exists with

```
$ docker images
```

This should list all the available images on your host

If you wish to push your code to your own repository on Docker hub, you can do so like this

```
$ docker push <username>/rlab:<tag>
```

Since this is not an open-source project (yet), the correct way to build onto the official xtnc/rlab image is to create a pull request to master branch.

3. Dockerfile specifications

   The xtnc/rlab image is built on top of the python:3.5.3-onbuild image. Although such base image is technically deprecated, it is used as a base image since it automatically runs a pip installation on root requirements.txt file

   The /researchs directory is set up to be a mountable volume. This allows the clear separation of research files and the research environment. Also, mounting a custom research directory at runtime will offer a more flexible and dynamic file structure.
   The /data directory is similar to /researchs directory, allowing runtime dynamic data from host machine.

   The template file is stored at /root/.ipython/profile_default/startup/ and must have the filename of startup.ipy

   The custom style sheet must be stored at /root/.jupyter/custom/ and must be named custom.css
   Alternatively a logo.png can be stored at the same directory for esthetics of the notebook

   Newly added python dependencies must be added to requirements.txt since it will be the first task of the base image to install them when building our own image

4. Template file specifications

   A few thing must always be in the template file, such as importing sys and appending the sys path with (..) . This is to tell all importing index to start one directory directly above. Since all research files are stored in /researchs which is hierarchically parallel to /data and /core, in order to be able to import files form data and core (where our custom written libraries are) we must set our import index one directory directly above.

For more reference please visit https://docs.docker.com/