StatisKit Documentation

Release 0.0.1

Pierre Fernique, Jean Peyhardi

Jun 24, 2020

CONTENTS

StatisKit is a collection of open source software designed to provide an environment for performing statistical analyses in *C++* or *Python*.

CHAPTER

ONE

GENERAL DOCUMENTATION

This documentation is dedicated to people using the **StatisKit** software suite. In this documentation, the following expressions shall have the following meaning:

User means any individual using **StatisKit** binary files published by maintainers.

Developer means any individual producing **StatisKit** binary files to be published by maintainers.

Maintainer means any individual publishing **StatisKit** binary files used by users.

Please, with regard to previous expressions, refers to the subsequent guides.

Note: While developer and maintainer roles seem to be separated, reading the maintainer guide is useful for developer, and *vice versa*.

More insights on the project can be found on the organization **GitHub** page

1.1 User Guide

Warning: Section under construction.

1.1.1 Test it !

In a first stage, you are not compelled to install **StatisKit** on your computer in order to discover its functionalities. Using **Docker** images, **Binder** servers and **Jupyter** notebooks, we are able to provide pre-installed interfaces with various examples.

Note: For more information refers to :

- The **Jupyter** documentation.
- The **Binder** documentation.
- The **Docker** documentation.

Online With Binder

To reproduce the various examples from a **Binder** server, follow this link.

On Your Computer With Docker

To reproduce the various examples with **Docker** use these images. After installing **Docker**, you can type the following command in a shell:

```
docker run -i -t -p 8888:8888 statiskit/python-statiskit:latest
```

Then, follow the given instructions.

Warning: By default, on some operating systems like Ubuntu, docker require to have administration rights. You can, for example, execute the preceeding lines after typing `sudo -i` if you are on Ubuntu or follow these instructions.

Note: If your port `8888` is already used, replace this number in these command lines and instructions given by another one (e.g., `8889`).

On Your Computer From a SSH Server

To reproduce the various examples from a **SSH** server, you can type the following commands in a shell:

```
ssh -L 8888:localhost:8888 <username>@<servername>
jupyter notebook --ip='*' --port=8888 --no-browser
```

Note: The username on the **SSH** server (resp. the **SSH** servername) is denoted in the following by `<username>` (resp. `<servername>`). Please replace it by the appropriate username (resp. servername).

Then, follow the given instructions.

Note: If your port `8888` is already used, replace this number in these command lines and instructions given by another one (e.g., `8889`).

1.1.2 Install it !

Prerequisites

In order to ease the installation of the **StatisKit** software suite on multiple operating systems, the **Conda** package and environment management system is used.

Note: For more information refers to the **Conda** documentation.

To install **Conda**, please refers to this page. Installers for:

- **Miniconda** are available on this page.
- **Anaconda** are available on this page.

Note: We recommend to:

- Follow the instructions given for the regular installation.
- Install **Miniconda** if you are only interested by **Statiskit**.
- Install **Miniconda 3** or **Anaconda 3** since the supported version of **Statiskit** is based on *Python 3*.

Recommanded Installations

The recommended installations rely on **Conda** meta-packages. To install the latest *Python 3* interface to **StatisKit**, type the following command line

```
conda create -n python-statiskit python-statiskit -c statiskit -c defaults --override-
↪channels
```

Then, to activate the `python-statiskit` environment, type the following command line

```
conda activate python-statiskit
```

1.1.3 Frequently Asked Questions

Here we try to give some answers to questions that regularly pop up or that could pop up on the mailing list.

1.2 Developer Guide

Warning: Section under construction.

1.2.1 Configure your Computer

In order to ease the development of the **StatisKit** software suite on multiple operating systems, the **Conda** package and environment management system is used. To install **Conda** refer to the section *Prerequisites*.

Once **Conda** is installed, you need to create a development environment called `statiskit` containing the meta-package `statiskit` on your computer. To do so, type the following command line

```
conda create -n statiskit-toolchain statiskit-toolchain -c statiskit -c defaults --
↪override-channels
```

Warning: On Windows OSes you must first download and install **Visual Studio Community 2017** (available on this page).

Then, to activate the `statiskit` environment for each build of **StatisKit** software suite components, type the following command line

```
conda activate statiskit-toolchain
```

Note: If you want a fine grained configuration, report to the *Frequently Asked Questions* section.

1.2.2 Contribute to a Repository

Warning: It is here assumed the `statiskit` environment has been installed and activated as detailed in the *Configure your Computer* section.

Note: This section heavily relies on the **devops-tools** program. For more information concerning the `github`, `travis_ci` and `appveyor_ci` commands, refer to their documentation.

Official repositories of **StatisKit** are currently hosted on GitHub. In order to contribute to an official repository of **StatisKit** we recommend to proceed as follows.

Note: In the following `<REPOSITORY>` denote the official repository name.

1. Fork the repository from the organization account to your personal account. If this repository is already forked on your personal account, you can skip this step. Otherwise, type the following command in your console

   ```
   github fork <REPOSITORY> --owner=StatisKit
   ```

2. Clone the repository from your personal account to your computer. If this repository is already cloned on your computer, you can skip this step. Otherwise, type the following command in your console

   ```
   github clone <REPOSITORY>
   ```

   Warning: After this step, it is assumed that your console working directory is the one of the local repository. Two remotes are available for this local repository:

   - The `upstream` remote pointing to the repository located on the organization account.
   - The `origin` remote pointing to the repository located on your personal account.

3. Activate Continuous Integration and Deployment (CI&D) services for the repository located on your personal account. This step is not mandatory but is recommended. To do so, type the following commands in your console

   ```
   travis_ci init --anaconda-label=main
   appveyor_ci init --anaconda-label=main
   ```

   Warning: To activate CI&D services, you need to have:

   - A Travis CI account.
   - A AppVeyor CI account.

3. Retrieve the latest code from the repository located on the organization account and push it together with your modifications to the repository located on your personal account. This step is particularly important if you skipped one of the first two.

   To do so, type the following commands in your console

```
git pull
git push
git pull upstream master
git push
```

   Warning: Before using these commands, it is better to make sure that there are no uncommitted changes nor untracked files on your local repository. To do so, type the following command in your console

```
git status
```

   If you want to suppress (permanently) all uncommitted changes, type the following command in your console

```
git reset --hard
```

   Moreover; if you want to suppress (permanently) all untracked files, type the following command in your console

```
git clean -fd
```

4. Work on your local repository. To work on a repository, an issue must first have been published.

   Warning: Issues must be published on the repository located on the organization account, not on your personal repository.

   To search for existing issues or creating new ones using your Web browser, type the following command

```
github issues --browser
```

   To display in your console current open issues, type the following command in your console

```
github issues
```

   To display in your console current open issues that are assigned to yourself (i.e., that you are currently working on), type the following command in your console

```
github issues --assigned
```

   In the following, we consider that an issue is identified by its number denoted by `<ISSUE>`. If this issue corresponds to:

   - a bug, the work must typically be situated on a branch named `hotfix_<ISSUE>` created from the `master` branch of the repository located on the organization account. Thus, type the following command in your console

```
github hotfix --issue=<ISSUE>
```

     Yet, if you do not have the necessary permissions to write on the repository located on the organization account, the branch must be created from the `master` branch of your personal account. To do so, type the following command in your console

```
github hotfix --issue=<ISSUE> --remote=origin
```

- an enhancement, the work must typically be situated on a branch named `feature_<ISSUE>` created from the `master` branch of the repository located on your personal account. Thus, type the following command in your console

  ```
  github feature --issue=<ISSUE>
  ```

  If the enhancement should be assigned to more than one developer (large ones), the branch must be create from the `master` branch of the organization repository. To do so, type the following command in your console

  ```
  github feature --issue=<ISSUE> --remote=upstream
  ```

Note: If the bug or the feature covers more than one issue, create a new issue referencing all those issues. In all those issues:

- add the `Duplicate of #<ISSUE>` comment,
- add the `duplicate` label.

Warning: If the branch name given by the *github hotfix* or *github fixture* commands corresponds to a remote branch, the remote will be set to the existing remote branch.

For more information concerning how to amend a repository, refer to the *Frequently Asked Questions* section. If this step has already been made once on your local repository, type one the following commands in your console

```
github start hotfix_<ISSUE>
```

or

```
github start feature_<ISSUE>
```

To see all available branches of your local repository, type the following command in your console

```
git branch
```

To see all available branches of all repositories, type the following command in your console

```
git branch -a
```

An easiest way if you have no concurrent branches is to use the following command line

```
github start
```

This command will ensure that you are currently working on the latest branch you edited using these commands.

Warning: If there are untracked files or uncommitted changes on your current local branch, this command will fail.

Similarly, to go back to the local `master` branch, type the following command

```
github end
```

Warning: If there are untracked files or uncommitted changes on your current local branch, this command will fail.

Note: At any point, to seek information about a particular issue using your Web browser, type the following command in your console

```
github issue <ISSUE> --browser
```

If you are currently working on a branch and want to seek information about the corresponding issue using your Web browser, type the following command in your console

```
github issue --browser
```

This is particularly helpful if you forgot the meaning of an issue number you were working on.

---

5. Retrieve the latest code from the repository located on the organization account and push it together with your modifications to the repository located on your personal account.

   To do so, type the following commands in your console

```
git pull
git push
git pull upstream master
git push
```

   Warning: Before using these commands, it is better to make sure that there are no uncommitted changes nor untracked files on your local repository. To do so, type the following command in your console

```
git status
```

   If you want to suppress (permanently) all uncommitted changes, type the following command in your console

```
git reset --hard
```

   Moreover; if you want to suppress (permanently) all untracked files, type the following command in your console

```
git clean -fd
```

6. Suggest to maintainers to incorporate your modifications into the `master` branch of the repository located on the organization account. To do so, type the following command in your console

```
github end --suggest
```

   Warning: If your local branch is at least one commit behind the `master` branch of the repository located on the organization account or is ahead of the corresponding branch on the repository location on your personal account, this command will fail.

1.2.3 Frequently Asked Questions

Here we try to give some answers to questions that regularly pop up or that could pop up on the mailing list.

How to Organize a Repository ?

It is important to have a common structure shared between all repositories. Yet, this is not currently a rule written in stone since this has not yet been really discussed. However, here is the current structure emerging from actual repositories

Directory	Description
`/`	Repository root directory
`/etc`	Essential files that need to be available for maintainers
`/etc/conda`	**Conda** recipes for generating **Conda** binaries
`/etc/docker`	**Docker** contexts for generating **Docker** images
`/doc`	Essential files that need to be available for documenters
`/share`	Essential files that need to be available for users
`/share/git`	**Git** sub-modules that need to be available for users
`/share/jupyter`	**Jupyter** notebooks that need to be available for users
`/src`	Essential files that need to be available for developers
`/src/cpp`	*C++* source code files
`/src/cpp/SConscript`	**SCons** configuration file for the *C++* library installation
`/src/py`	*Python* source code files
`/src/py/wrapper`	*Boost.Python* source code for interfacing the *C++* library with *Python*
`/src/py/wrapper/SConscript`	**SCons** configuration file for the *C++*/*Python* binding library generation
`/test`	Test files
`/travis.yml`	**Travis CI** configuration file
`/.travis.yml`	A symbolic link to the **Travis CI** configuration file
`/appveyor.yml`	**Appveyor CI** configuration file
`/SConstruct`	**SCons** general configuration file

How to Configure my IDE (Integrated Development Environment) ?

For developers, it can be convenient to use an IDE. Currently, each repository can be used with:

- **Sublime Text**. To add a **Sublime Text** build system compatible with **StatisKit** repositories, use the following command

  ```
  build_system sublime_text
  ```

  Moreover, with **Sublime Text**, it is recommanded to use the following addons:

  - **Package Control**, see this *page <https://packagecontrol.io>* for more details.
  - **Terminal**, see this *page <https://packagecontrol.io/packages/Terminal>* for more details.
  - **Git**, see this *page <https://packagecontrol.io/packages/Git>* for more details.
  - **ProjectManager**, see this *page <https://packagecontrol.io/packages/ProjectManager>* for more details.

Note: Any other IDE build system proposal is welcome.

How to Speed Up Build Time ?

For projects using **SCons** or developers using build systems in IDEs (see the *How to Configure my IDE ?* section), it is possible to speed up the build time by parallelizing most of *C++* compilations. This is usually done using the `-j<CPU_COUNT>` flag with **SCons**: For example

```
scons -j6
```

will use, when possible, 6 concurrent compilations.

Warning: It is not recommended to set `<CPU_COUNT>` to a value superior to your number of processors. From a console, you can see yout number of processors by typing the following command line in a *Python* interpreter:

```
import multiprocessing
multiprocessing.cpu_count()
```

To avoid using the `-j<CPU_COUNT>` flag each time, you can type the following command line in your console:

```
cpu_count
```

This will automatically set the number of concurrent compilations to your number of processors minus one. You can manually specify the number of concurrent compilations using the `--number <CPU_COUNT>` flag.

What is the *C++* Style Guide ?

Warning: Section under construction. Until further notice, please use the Google *C++* style guide

- A repository should contain at most 1 *C++* library.
- The *C++* library source code must be located in the `scr/cpp` directory.
- To install headers of the *C++* library, a developer should use the following command in the repository root

  ```
  scons cpp-dev
  ```

- To generate and install the *C++* library binaries, a developer should use the following command in the repository root

  ```
  scons cpp-lib
  ```

- The following command

  ```
  scons cpp
  ```

  should be equivalent to the following commands

  ```
  scons cpp-dev
  scons cpp-lib
  ```

- If the *C++* library is interfaced in any other languages (e.g., *Python* or *R*), the wrappers should be generated using the following command

```
scons autowig
```

Note: In this case, guidelines proposed in the **AutoWIG** documentation are of most importance.

What is the *Python* Style Guide ?

Warning: Section under construction. Until further notice, please use the Google *Python* style guide.

- A repository should contain at most 1 *Python* package.
- The *Python* package source code must be located in the `scr/py` directory.
- To install the *Python* package, a developer should use the following command in the repository root

```
scons py
```

Note: If this package is an interface of a *C++* library, this command should also generate relevant binaries.

How to Update the Development Environment ?

If you need to update your development environment, type the following command line in your console

```
conda update --all --no-pin -c statiskit/label/develop -c statiskit -c defaults --
↪override-channels
```

Warning: In this case, the development environment must first be activated

In the worst case scenario, you can first uninstall your development environment and re-install it. To do so, type the following command lines in your console

```
conda env remove -n statiskit -y
conda clean --all -y
conda create -n statiskit statiskit -c statiskit/label/develop -c statiskit -c␣
↪defaults --override-channels
```

Warning: In this case, the development environment must first be deactivated

1.3 Maintainer Guide

Warning: Section under construction.

1.3.1 Configure Your Computer

In order to ease the deployment of the **StatisKit** software suite on multiple operating systems, the **Conda** package and environment management system is used. To install and configure **Conda** refer to the section *Configure your Computer*.

1.3.2 Create a New Repository

Warning: It is here assumed the:code:*statiskit-dev* environment has been installed and activated as written in Section *Configure your Computer*. This section heavily relies on the **devops-tools** program. For more information on the `github`, `travis_ci` and `appveyor_ci` commands, refer to their *documentation <http://devops-tools.rtfd.io>*.

Official repositories of **StatisKit** are currently hosted on GitHub. In order to create an official repository of **StatisKit** we recommend to proceed as follows.

Note: In the following `<REPOSITORY>` denote the official repository name.

1. Initialize the repository on the organization account. To do so, type the following command in your console

```
github init <REPOSITORY> --owner=StatisKit --license=apache-2.0
```

2. Clone the repository from the organization account to your computer. If this repository is already cloned on your computer, you can skip this step. Otherwise, type the following command in your console

```
github clone <REPOSITORY> --owner=StatisKit
```

   Warning: After this step, it is assumed that your console working directory is the one of the local repository.

3. Activate Continuous Integration and Deployment services for your repository. Contrarily to user repositories, this step is mandatory for organization account's repositories. To do so, type the following commands in your console

```
travis_ci init --anaconda-label=develop
appveyor_ci init --anaconda-label=develop
```

3. Populate the repository with relevant files.

   Warning: Until now, the repository structure has not been clearly set. More information can be gathered in the *Frequently Asked Questions* section. A package is yet under consideration to propose command lines to simplify the process (e.g., `layout init`).

4. Retrieve the latest code from the repository located on the organization account and push your modifications to the repository located on the organization account.

   To do so, type the following commands in your console

```
git pull
git push
```

   Warning: Before using these commands, it is better to make sure that there are no uncommitted changes nor untracked files on your local repository. To do so, type the following command in your console

```
git status
```

   If you want to suppress (permanently) all uncommitted changes, type the following command in your console

```
git reset --hard
```

   Moreover; if you want to suppress (permanently) all untracked files, type the following command in your console

```
git clean -fd
```

Note: For more information concerning naming conventions and places for files specific to further repository developments, refer to the *Frequently Asked Questions* section.

1.3.3 Frequently Asked Questions

Here we try to give some answers to questions that regularly pop up or that could pop up on the mailing list.