Python & R Installation & Usage Guide

1 Python

1.1 Installation

To get started with Python, the following are needed:

- Python
- Package (and environment) manager
- IDE/Notebook editor

First, you can download the latest version of Python from the Python website. When installing Python, there are several “Optional Features” available. Any features that are checked as default should be installed, but you should also make sure that ‘pip’ is one of the features that is checked.

Next, there are several “Advanced Options” that are available to select in the installation process. Again, the defaults should remain checked, but in addition to these, the “Add Python to environment variables” option needs to be checked. This should mean that Python is setup properly on your local machine.

Following this, you can install miniconda for package and virtual environment management.

There are several options for IDEs or notebook editors, but if you have no previous experience with programming of any kind, I recommend JupyterLab first. If you have experience with programming generally, but not R, I suggest VS Code. Finally, if your experience with programming comes from R (and presumably RStudio), I would recommend either RStudio (particularly if you plan to continue using R), or Spyder.

1.2 Setting up Virtual Environments

A virtual environment is a tool for isolating projects such that the packages and all dependencies for the project do not interfere with any other projects. You can use specific versions of a package without that changing the version of the same package you are using in another project, for example. This is an important part of the development process in Python, and it helps ensure that a project is reproducible. To create a new environment, open a terminal window, and you can create and then active your new project environment, with the following:

```
conda create --name my-project

conda activate my-project
```

The Conda documentation has a good explanation for how to set up and manage a virtual environment.

1.3 Installing Packages

There are two main ways to manage packages – pip and conda. You can install packages using either, and if you have activated your virtual environment, this will install your package in that environment only.

If you wanted to install the `fingertips_py` package, you need to use pip, using `pip install fingertips_py`.

While `fingertips_py` isn't available using conda, a lot of packages are available using both pip and conda. If you wanted to install `scipy` using conda (if you wanted to use pip, you would use the same code as above, replacing `fingertips_py` with `scipy`), you would use `conda install scipy`.

The Conda documentation has a good explanation for how to install packages.

In order to take a snapshot of a project's dependencies, you can use `conda env export > environment.yml`. This will make your project reproducible for others.

2 R

2.1 Installation

Getting up and running with R is a little simpler than with Python, and the key to getting R working well on SCW laptops appears to be making sure everything is installed on your C drive.

To get started with R, the following are needed:

- R & RStudio
- RTools

First, you can download and install the latest version of R from CRAN, and the latest version of RStudio from the Posit website. Following this, you will also need RTools, which can be downloaded from CRAN.

With R, RStudio, and RTools all installed, you should be able to get started using R.

2.2 Setting up Virtual Environments

Using a virtual environment is less of a necessity in R compared with Python, but it is still good practice! Managing environments can be handled directly in the Console in RStudio, using the `renv` package.

`renv` must first be installed using `install.packages("renv")`. Having installed `renv`, you can initialise (and activate) a new environment when working on a new RStudio Project using `renv::init()`.

2.3 Installing Packages

Packages can be installed using either `install.packages()` or `renv::install()`, and they can be updated using `update.packages()` or `renv::update()`.

In order to take a snapshot of your project's dependencies, use `renv::snapshot()`.

For more information about environment and dependency management in R, the `renv` documentation is the best place to start.