Andre Brezski 04.02.2021

Coding tasks for 04.02.2021

Topic: Introduction to Conda

1. Install Miniconda on your working device

Windows:

```
start /wait "" Miniconda3-latest-Windows-x86_64.exe /InstallationType=JustMe
/RegisterPython=0 /S /D=%UserProfile%\Miniconda3
```

Linux:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

MacOS:

```
bash Miniconda3-latest-MacOSX-x86_64.sh
```

2. Try to update your Conda installation

→ Should be done from time to time

Solution:

```
conda update conda
```

3. Create a Conda environment with python 3.9

→ Conda environments can also be installed without calling any packages

→ Mostly a specific python-version is beneficial (requirement for some tools)

Solution:

```
conda create -n myenv python=3.9
```

4. Write an environment.yml with the dependencies numpy, pandas and python 3.6

→ Useful if an environment with a lot of specific packages (and versions) is needed

→ Can also be used to share or reproduce certain environments

Solution:

```
name: stats
dependencies:
  - python=3.6
  - numpy
  - pandas
```

Creating an environment from a yaml-file:

```
conda env create -f environment.yml
```

5. Activate the environment

→ For using the environment it has to be activated on the shell

→ If used on the cluster:

- Specific environment has to be activated when using sbatch
- It is then automatically used when running the job (no open shell needed)

Solution:

```
conda activate myenv
```

6. View all installed packages

→ Shows a list of all installed packages with version numbers

→ Also useful to check used tools/packages for methods

Solution:

```
conda list -n myenv
```

7. Install scipy directly in the environment

→ Specific version and channel can also be called inside the command

→ Requirements for the package are also installed

→ Beware: Previous installed packages can be up- or downgraded in this process!

Solution:

```
conda install -n myenv scipy
```

8. Check if scipy was installed

→ Easy solution is to look inside the list of all installed packages

Solution:

```
conda list -n myenv
```

9. Try to update scipy

→ Sometimes conda initially does not install the actual version

→ Especially when no specific version is provided in the installation command

→ Checking for updates can be important for packages to run correctly!

Solution:

```
conda update -n myenv scipy
```

10. Remove scipy from your environment

→ Packages which are not needed can be easily removed

→ Also useful if certain packages exclude each other

Solution:

```
conda remove -n myenv scipy
```

11. Export the environment into an environment.yml-file

→ Exporting an environment to a yaml-file makes it easily shareable

→ Also useful as backup for reproducing certain environments
after changing something

Solution:

```
conda env export > environment.yml
```

Creating an environment from a yaml-file:

```
conda env create -f environment.yml
```

12. Remove the environment

→ Conda environments can get quite large

→ > 2 GB per environment depending on installed packages/tools

→ Export unused environments as yaml-file (for later usage) and delete it

Solution:

```
conda remove --name myenv –all
```

Some additional useful information

→ You can prevent packages from updating:
- create a file named „pinned“ in the conda-meta directory
- add file names with the version they should stay at

→ It is possible to use more then one environment in a single script:
- activating a new environment normally deactivates the other environment
- sometimes you need to utilize more then one environment
- e.g. for tools with different dependencies

Solution: `conda activate --stack myenv`

- stacking environments allows usage of packages from both

Conclusion

Pros:

→ Conda is a relatively simple way to create customized environments

→ The environments can also be used on the cluster without any extra work

→ This allows installation of tools/packages without sudo rights

→ Easy shareable environment-setups with yaml-files

Cons:

→ certain tools/packages are not available for conda (installation with pip or manual)

→ sometimes getting the right versions and dependencies of packages can get lengthy

(→ originally developed for python)