## **Virtual Environments Notes**

### Renv (Reproducible Environments) in R

Want to start your project on the right foot? Check the option to start a new project with Renv or in the

If you are using your Rstudio not the server, you need to install renv

install.packages("renv")

You may also choose Tools>Project Options>Environments and check “use renv for this project”

If starting a project from Git you may run:

renv::init( )

Let’s check the new files that we have…

Look at the git ignore

Look at the renv.lock file

Let’s say we reworked our script and suppose we would need sdcMicro. Let’s update and include library (sdcMicro) in our script and save it. Then, install sdcmicro:

renv::install(“sdcMicro”)

Alright, now that the installation is completed, we can save, and take a snapshot:

renv::snapshot( )

This action will update the lock file, and we will see sdcMicro and all its dependencies included.

Let’s check. If updated, you should be good to go. If your attempts to update packages introduced new problems, you may run renv::restore() to revert to the previous state as encoded in the lockfile.

Use .libPaths( ) to confirm where package installations are located.

### Venv Using pip

#### Open the Terminal in VS Code

**Checking our version:**

Python --version

Venv comes installed in Python 3 and above.

**Checking what is in the system:**

pip list

A long list of packages, right? So assume we are working with only a few and want to call the exact versions and dependencies we need. That can be accomplished with venv.

**Create a new environment:**

* Create a new folder, your project folder. I will use EDS213. Make sure you set the path to this folder.
* Note: The environment will be created in the current version of Python that you are running (in Conda we can specify the version we want).

To create the environment: (second venv is the name of the environment)

python3 -m venv venv213

* The -m flag makes sure you are creating a pip that is tied to the active Python executable

**Time to activate it:**

source venv213/bin/activate

You can tell it is activated because it shows (venv213) in the prompt.

Let’s check which packages are there with a new pip list

Nothing, right? Only setup tools, and pip). Nothing to worry about, it should be this way! Let’s proceed.

**Install packages:**

We will be installing two packages for this exercise.

First:

pip install Numpy

And then:

pip install Pandas (this should take a little longer)

Another pip list

Alright, the packages and dependencies installed are right there!

**Export and allow future replication of the environment:**

Let’s save the packages and dependencies we have after the installs.

pip freeze

That should be stored in a requirements.txt file

requirements.txt

So let’s get it redirected to the required file:

pip freeze > requirements.txt

Question: This file won’t leave inside the venv folder, but rather in the project root folder any idea why?

Well, you only need that file to reproduce the environment. And the venv should be should throw away and be able to rebuild easily! So, do not include any project file in that folder and treat that as disposable after the pip freeze

To double-check if all is good, we can run the following command:

cat requirements.txt

This file should be included in your research compendium to let others reinstall your packages and dependencies as needed.

**Deactivate**

If you are done with that, you should deactivate that environment by typing:

deactivate

Then, you will see you no longer have the environment we created in our prompt.

If you are willing to delete the environment altogether, you should delete the directory for the virtual environment

Remove folder:

rm -rf venv213/

**Reusing the Requirements**

*Create a new project folder to reuse the requirements*

mkdir my-project

*Create a virtual env for it*

python3 -m venv my-project/venv

*Activate it*

source my-project/venv/bin/activate

*Install required packages*

pip install -r requirements.txt

* Attention! Never include project files in the venv folder.
* Do not commit your venv file to the environment itself to source control (git ignore)
* You may install more packages and update the requirements.txt with the pip freeze command
* You should commit /share only your requirements.txt file. That is all that others and your future self need to recreate the environment.
* The environment should be something you should throw away and be able to rebuild easily.
* Make sure to deactivate when done using it.

### Environments in Conda

**Checking what is in the system:**

conda list

To create the environment:

conda create --name env213

proceed ([y]/n)? y

Time to activate it

conda activate env213 only work on conda 4.6 and later versions. For Conda versions prior to 4.6, run:

* Windows: activate or Linux and macOS: source activate

You can tell it is activated because it shows (venv213) in the prompt.

Let’s check which packages are there with a new conda list env213

Empty, right? Nothing to worry about, it should be this way! Let’s proceed.

**Install packages:**

We will be installing two packages for this exercise.

First:

conda install Numpy

proceed ([y]/n)? y

And then, one more package:

conda install Pandas

proceed ([y]/n)? y

Now check which packages are in the specific environment we are working on:

conda list

Alright, the packages and dependencies installed are right there!

**Export and allow future replication of the environment:**

Let’s save the packages and dependencies we have after the installs.

conda list --export

That should be stored in a environments.yml file

So let’s get it redirected to the required file:

conda list -e > environment.yml

This file should be included in your research compendium to let others reinstall your packages and dependencies as needed.

**Deactivate it:**

If you are done with that, you should deactivate that environment by typing:

conda deactivate only works on conda 4.6 and later versions. For conda versions before 4.6, run:

Windows: deactivate or Linux and macOS: source activate

Then, you will see you no longer have the environment we created in our prompt.

If you are willing to delete the environment altogether, you should delete the directory for the virtual environment

**Back to base we can create a new environment based on the .yml packages and dependencies by running (this is noted on top of the yml file):**

conda create --name my-env --file environment.yml

proceed ([y]/n)? y

**Activate it:**

conda activate my-env (or see above if conda version < 4.6)

proceed ([y]/n)? Y

**Check if packages are there:**

conda list

**Remember to deactivate it when done:**

conda deactivate (or see above if conda version < 4.6)

Check all your environments conda env list

More info: <https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html>