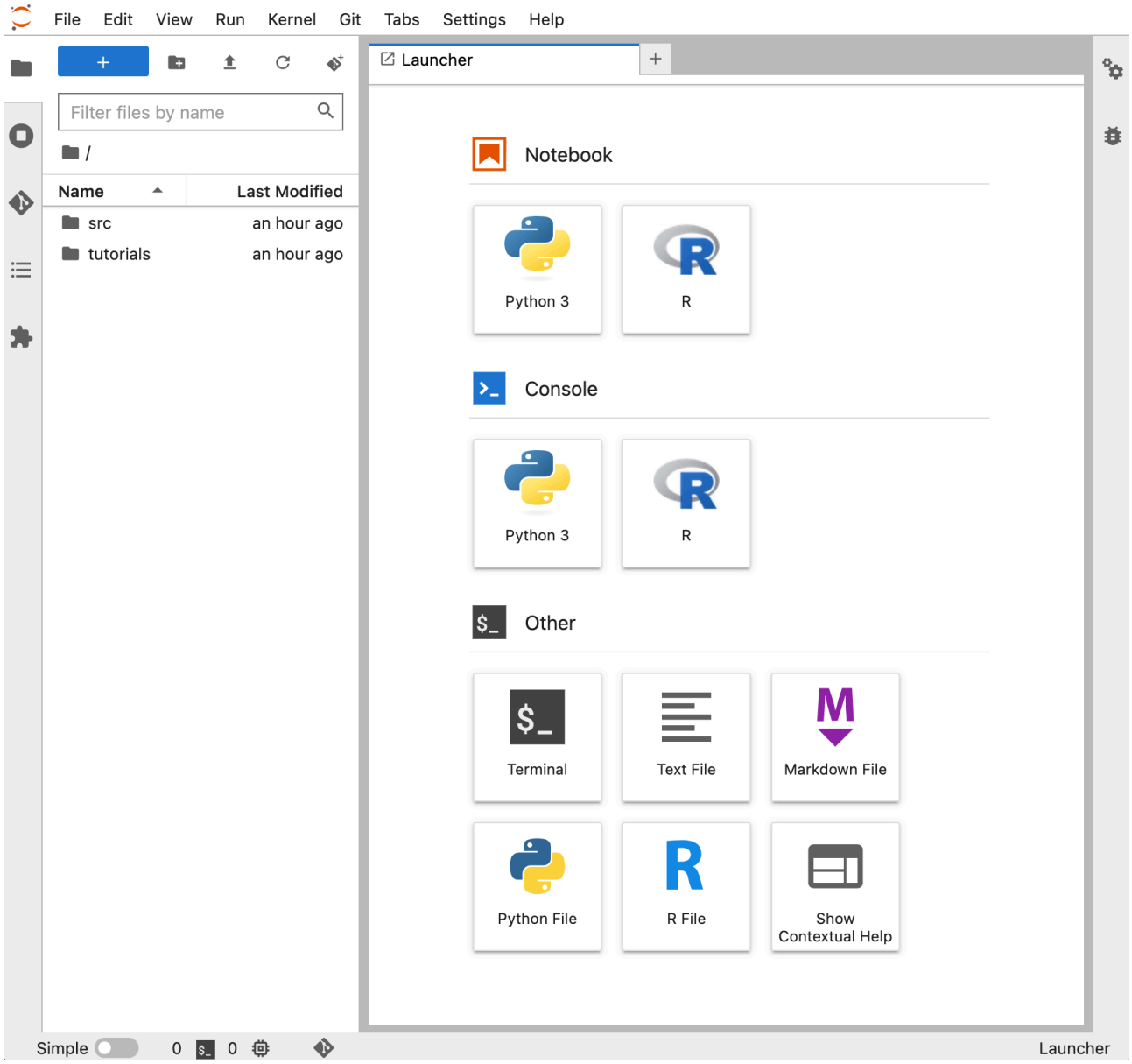
Setting Up an R Environment in Vertex Workspace

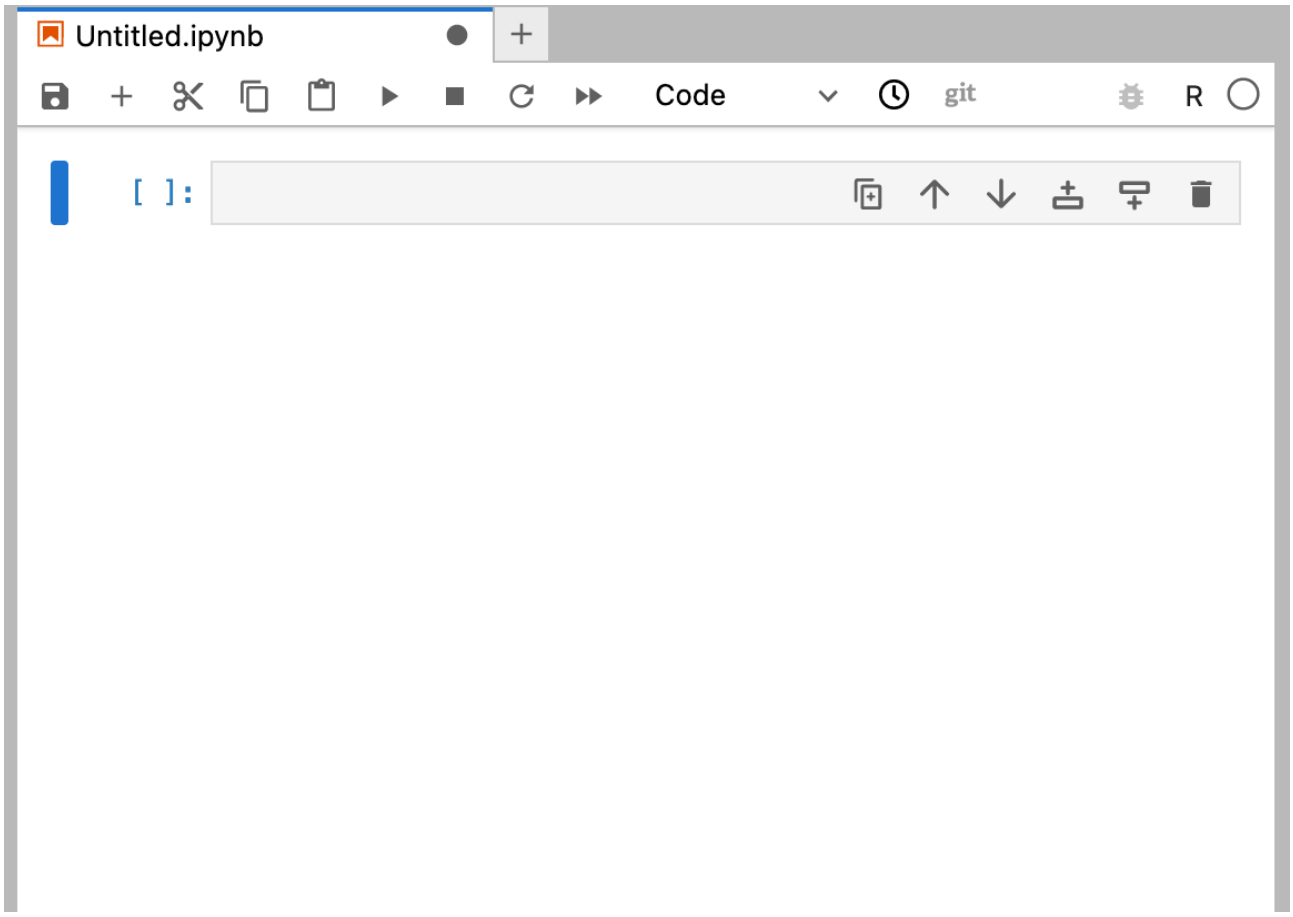
Intro to JupyterLab

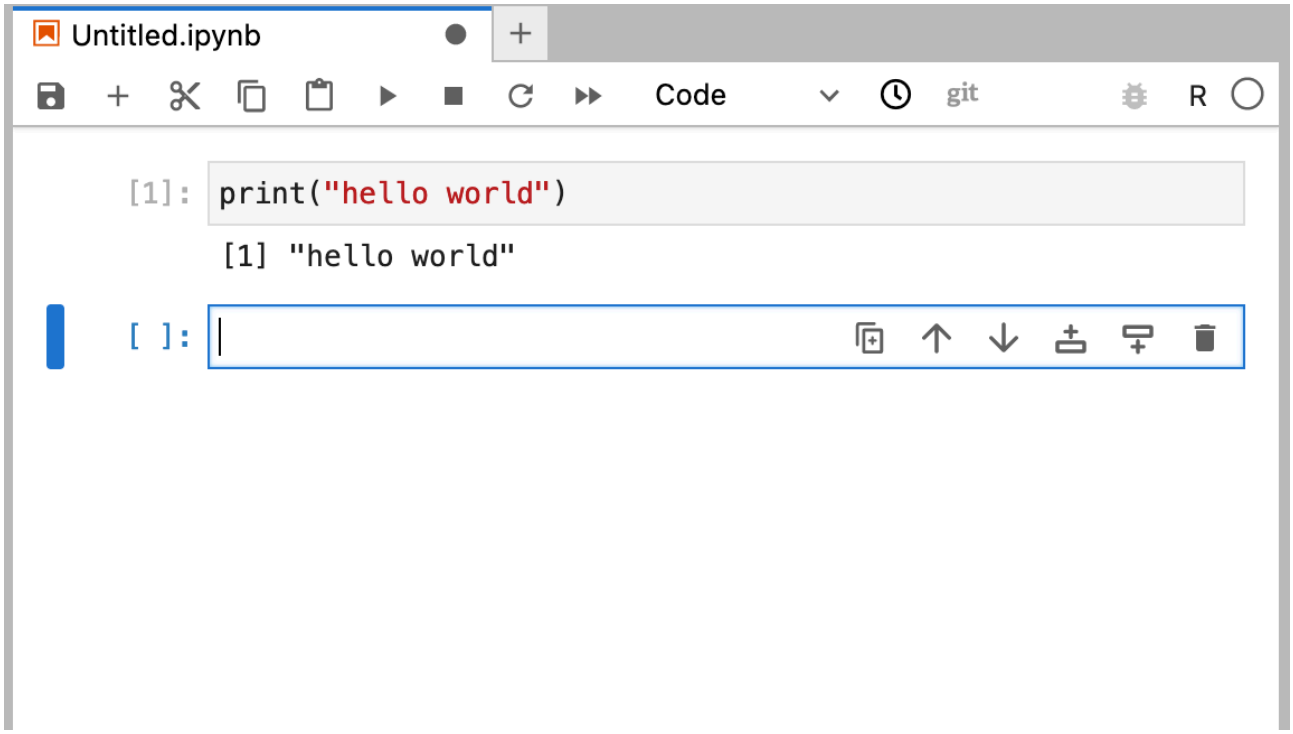
The following is a quick intro to get you started using R in Vertex Workspace. Vertex uses the JupyterLab IDE. JupyterLab was designed originally with Python in mind, but has since been adapted to accommodate R relatively comfortably. JupyterLab allows users to run code in interactive notebooks - documents that allow users to write and run code in one space, which allows for a much more fluid workflow and easier reporting.

As a quick intro, open up your Vertex Workspace. You should see the following:



If you click on the R logo under “Notebook”, you should see something that looks like this:





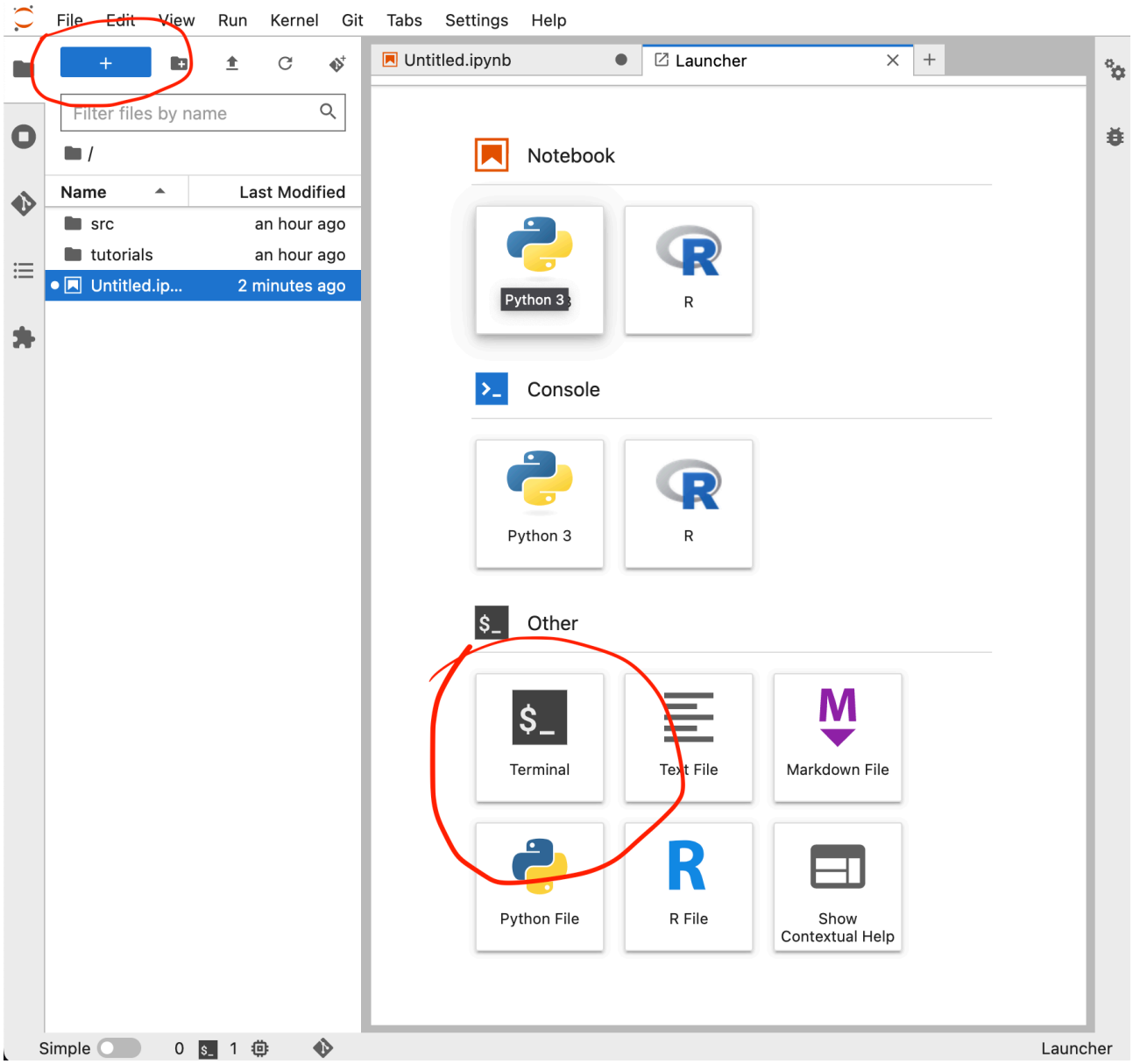
This is a notebook. The box at the top of the page is a code cell - if you write the classic “hello world” script and either click the “play” button in the toolbar at the top of the screen, or hit “shift + enter” with the code cell highlighted, you’ll run the cell like so:

If you need a more in-depth guide into the functionality of notebooks, take a look at this following (just replace all the python with R)

Installing Libraries/Packages

The majority of the functionality shouldn't trouble R users, but there is one key difference with using R in this environment to other settings: Anaconda. Anaconda is a package manager that was, again, designed for use with Python. It can, however, be used to install R and the vast majority of R libraries too. The version of R in the Vertex JupyterLab environment needs to be managed using Anaconda - that means installing all libraries using the conda command line interface (CLI). That also means never installing any R libraries through CRAN - it won't usually work, and even if it does, it will have unexpected consequences. Don't do it - you've been warned!

To get started with the conda CLI, you'll need to open a terminal. Go to the launcher - which you can open by clicking this plus at the top of the sidebar on the left of the screen and then clicking on the terminal icon, which is at the bottom of the launcher page under "other":

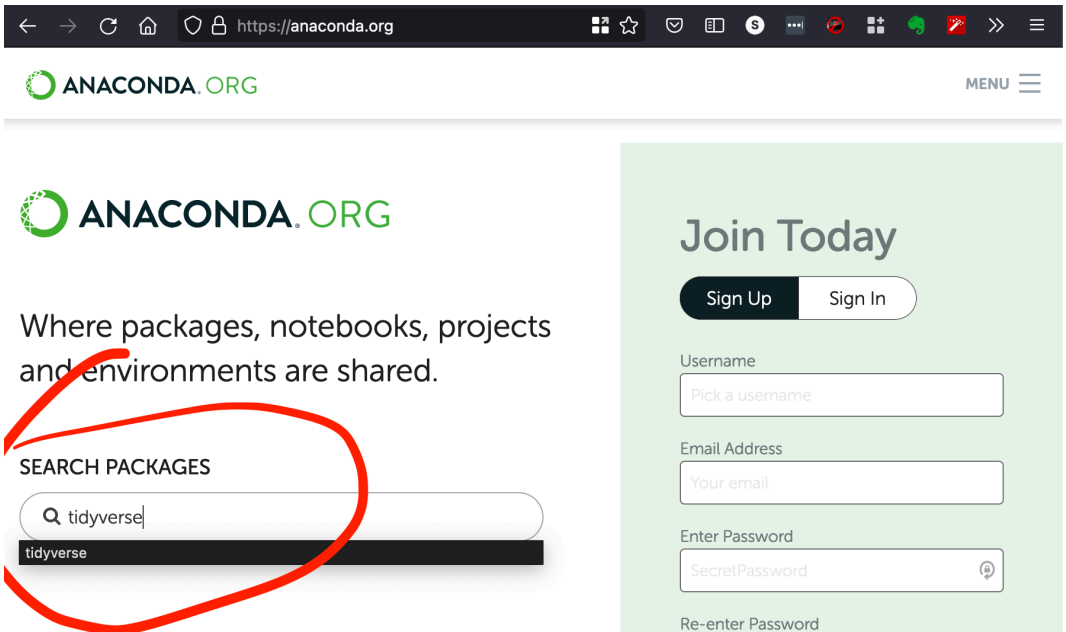


Once in a terminal, you can run anaconda commands by typing “conda” and then the requisite keywords that signify the function you want anaconda to execute. Just typing conda will return some documentation to give you a quick prompt:

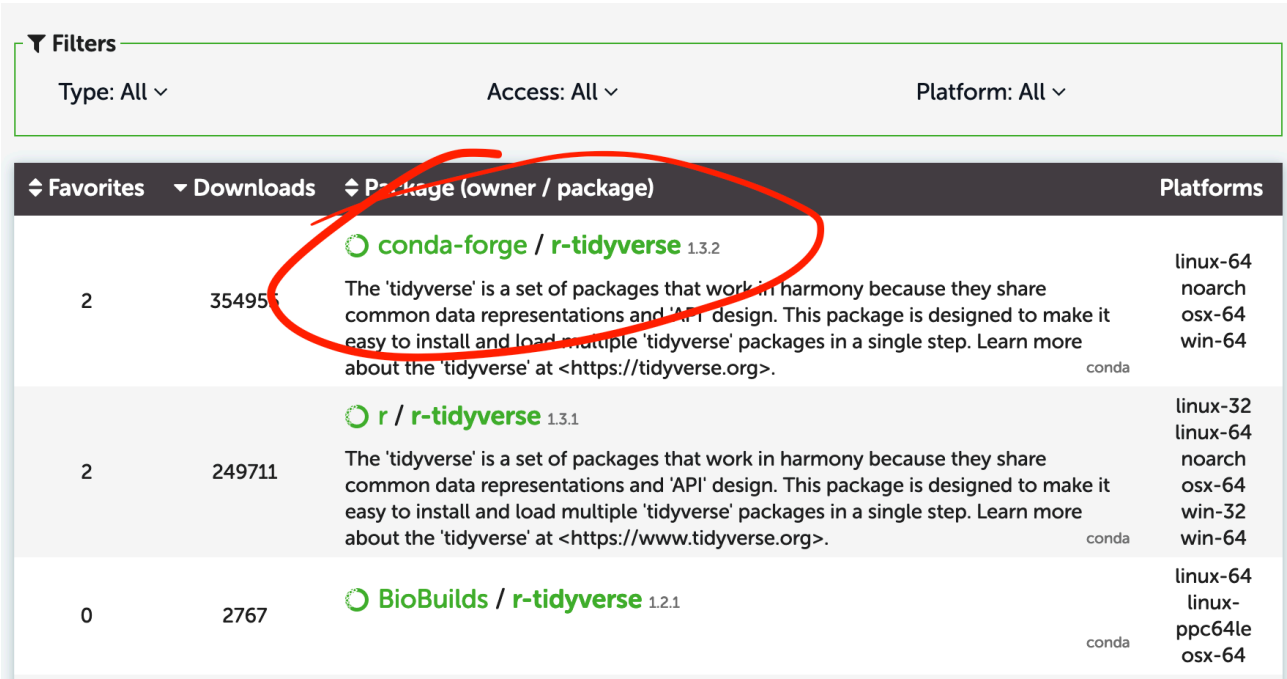
Untitled.ipynb | Terminal 1

```
(base) jupyter@test-tetst-testy-test:~$ conda
usage: conda [-h] [-V] command ...

conda is a tool for managing and deploying applications, environments a
nd packages.

Options:

positional arguments:
  command
    clean        Remove unused packages and caches.
    compare      Compare packages between conda environments.
    config       Modify configuration values in .condarc. This is
                 modeled after the git config command. Writes to the
                 user .condarc file (/home/jupyter/.condarc) by
                 default. Use the --show-sources flag to display all
                 identified configuration locations on your computer.
    create       Create a new conda environment from a list of
                 specified packages.
    info         Display information about current conda install.
    init         Initialize conda for shell interaction.
    install      Installs a list of packages into a specified conda
                 environment.
    list         List installed packages in a conda environment.
    package      Low-level conda package utility. (EXPERIMENTAL)
    remove       Remove a list of packages from a specified conda
                 environment.
    rename       Renames an existing environment.
    run          Run an executable in a conda environment.
    search       Search for packages and display associated
                 information.The input is a MatchSpec, a query
                 language for conda packages. See examples below.
```

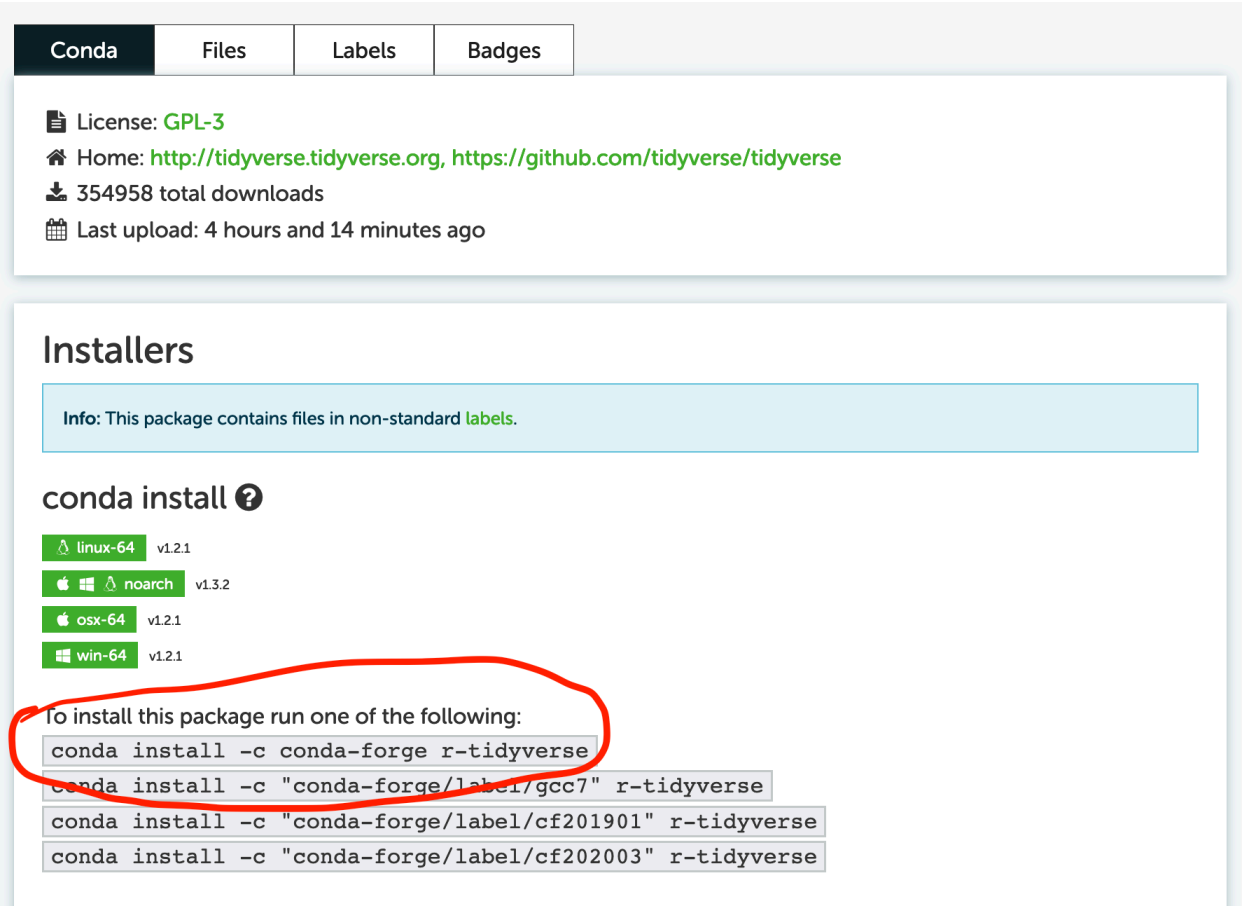
To install an R library, you first need to find it in the anaconda package library. That’s quickly done by searching on anaconda.org:



Search for the package you need and you'll be presented with a list of packages that match that keyword - usually there'll be an "conda-forge" package at the top of the search list, which should be your default choice:



Click on that package and you'll be presented with a page that includes terminal commands to install the package. Copy paste the command into your terminal and you're good to go!



```
Untitled.ipynb                    Terminal 1

(base) jupyter@test-tetst-testy-test:~$  conda install -c conda-forge r-tidyverse
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/conda

  added / updated specs:
    - r-tidyverse


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _r-mutex-1.0.1             |      anacondar_1           3 KB  conda-forge
    binutils_impl_linux-64-2.36.1|       h193b22a_2        10.4 MB  conda-forge
```

Anaconda will check for any requirements and conflicts with other installed packages, and then collect a list of packages to install - hit y if you're happy with the options.

```
Untitled.ipynb                    Terminal 1

  r-xfun             conda-forge/linux-64::r-xfun-0.33-r41h7525677_1 None
  r-xml2             conda-forge/linux-64::r-xml2-1.3.3-r41h7525677_1 None
  r-yaml             conda-forge/linux-64::r-yaml-2.3.5-r41h06615bd_1 None
  sed                conda-forge/linux-64::sed-4.8-he412f7d_0 None
  sysroot_linux-64   conda-forge/noarch::sysroot_linux-64-2.12-he073ed8_15 None
  tktable            conda-forge/linux-64::tktable-2.10-hb7b940f_3 None
  xorg-kbproto       conda-forge/linux-64::xorg-kbproto-1.0.7-h7f98852_1002 None
  xorg-libice        conda-forge/linux-64::xorg-libice-1.0.10-h7f98852_0 None
  xorg-libsm         conda-forge/linux-64::xorg-libsm-1.2.3-hd9c2040_1000 None
  xorg-libx11        conda-forge/linux-64::xorg-libx11-1.7.2-h7f98852_0 None
  xorg-libxext       conda-forge/linux-64::xorg-libxext-1.3.4-h7f98852_1 None
  xorg-libxrender    conda-forge/linux-64::xorg-libxrender-0.9.10-h7f98852_1003 None
  xorg-libxt         conda-forge/linux-64::xorg-libxt-1.2.1-h7f98852_2 None
  xorg-renderproto   conda-forge/linux-64::xorg-renderproto-0.11.1-h7f98852_1002 None
  xorg-xextproto     conda-forge/linux-64::xorg-xextproto-7.3.0-h7f98852_1002 None
  xorg-xproto        conda-forge/linux-64::xorg-xproto-7.0.31-h7f98852_1007 None


Proceed ([y]/n)? y
```

And you're done, library installed! Well, not quite. See the next topic - environments:

Environments

Packages may be easy to install, but they can be tricky as you keep adding more and more - the requirements become more complex, and are more likely to conflict with other existing packages. As such, it's best practice to isolate all of the packages you need for different projects into different environments. Fortunately, Anaconda makes this easy.

You can create a new conda R environment by running the following:

```
conda create -n testEnv r-essentials r-base
```

The -n tells conda the following word is the environment name - so the above environment will be called "testEnv". All the commands following the environment name are packages that conda

should install in your new environment - so we've asked for "r-essentials" and "r-base". Running this line of code will present you with a prompt asking you if you wish to proceed installing all the specified requirements, much like the one we saw when installing packages. Enter "y" and your environment will be created:

```
Untitled.ipynb                    Terminal 1

  xorg-libxrender     conda-forge/linux-64::xorg-libxrender-0.9.10-h7f98852_1003 None
  xorg-libxt          conda-forge/linux-64::xorg-libxt-1.2.1-h7f98852_2 None
  xorg-renderproto    conda-forge/linux-64::xorg-renderproto-0.11.1-h7f98852_1002 None
  xorg-xextproto      conda-forge/linux-64::xorg-xextproto-7.3.0-h7f98852_1002 None
  xorg-xproto         conda-forge/linux-64::xorg-xproto-7.0.31-h7f98852_1007 None
  xz                  conda-forge/linux-64::xz-5.2.6-h166bdaf_0 None
  zeromq              conda-forge/linux-64::zeromq-4.3.4-h9c3ff4c_1 None
  zipp                conda-forge/noarch::zipp-3.9.0-pyhd8ed1ab_0 None
  zlib                conda-forge/linux-64::zlib-1.2.12-h166bdaf_4 None
  zstd                conda-forge/linux-64::zstd-1.5.2-h6239696_4 None


Proceed ([y]/n)? y


Downloading and Extracting Packages
r-boot-1.3_28        | 633 KB    |                                                    |
r-boot-1.3_28        | 633 KB    | ################################################## |
r-boot-1.3_28        | 633 KB    | ################################################## |
 100%
debugpy-1.6.3        | 2.0 MB    |                                                    |
debugpy-1.6.3        | 2.0 MB    | ################################################## |
debugpy-1.6.3        | 2.0 MB    | ################################################## |
 100%
```

To use this new environment, we need to activate it by running:

```
conda activate testEnv
```

Always double check that you're using the right environment by looking at your terminal cursor - it should be prefixed with the name of the current environment you're using:

```
Untitled.ipynb                    Terminal 1

(base) jupyter@test-tetst-testy-test:~$ conda activate testEnv
(testEnv) jupyter@test-tetst-testy-test:~$
```

If you ever forget the names of any of the environments you've created, simply run:

```
conda env list
```

And conda will list all the environments you've created.

With your environment activated, you can go ahead and install any packages you want using the above instructions. But, before you can use the environment for the first time JupyterLab needs to know it exists. First, you'll need to install the following dependencies (being sure you still have your environment activated - check the cursor prefix):

```
conda install -c conda-forge r-devtools r-irkernel
```

Then, open an R prompt by typing "R" in your terminal like so (and triple check you've got your environment activated):

```
Untitled.ipynb    Terminal 1

(testEnv) jupyter@test-tetst-testy-test:~$ R

R version 4.1.3 (2022-03-10) -- "One Push-Up"
Copyright (C) 2022 The R Foundation for Statistical Computing
Platform: x86_64-conda-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Then run the function:

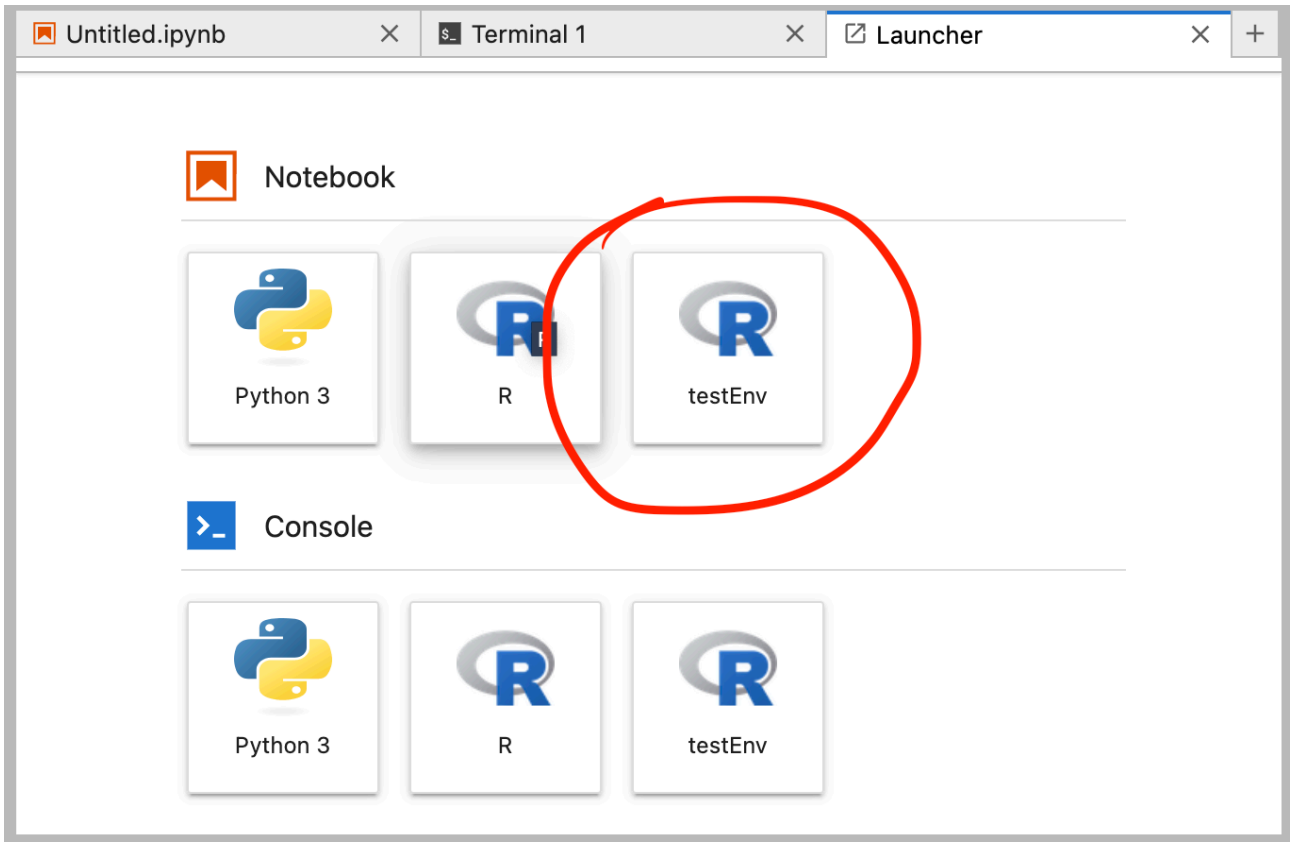
```
IRkernel::installspec(
        name="[your environment name]",
        displayname="[name that will appear under launcher icon]"
)
```

Replacing [your environment name] with the name of the environment you've created and [name that will appear under launcher icon] with a name of your choice (I usually just use the environment name again):

```
Untitled.ipynb    Terminal 1

> IRkernel::installspec(name="testEnv", displayname="testEnv")
>
```

(Like all great R libraries, this function gives absolutely no indication it's done anything, and I've never seen it throw an error - the only proof it's done anything is "in the pudding" so to speak!)

If you then open a launcher, you should hopefully see a new R icon with your new environment - clicking on this icon will open a new notebook, with your new R environment running under the hood:



And there you have it! You should now be set to run wild with R in Vertex Workbench!