R Setup and Testing

This guide will help you get your apprentices set up and ready for R in your Module 3 delivery session, as well as check for some common issues that can arise. It should only take 5-10 minutes to complete with the apprentices (plus some preparation on your part), and will help make sure your R session runs smoothly. I suggest you go through this process in the previous delivery session, to give plenty of time to address any issues before getting to R.

Unless you're confident in setting up R, don't try to debug errors in this process yourself (unless it's something obvious, like a typo in a command). Just get screenshots of any error messages and note down what happened, then carry on with your session and ask for help in the df-team Slack channel afterwards. The point of checking the setup well in advance is that there's then no rush on finding a solution!

Preparation: Choose from your installation options

There are several ways to run R, and what you'll pick will depend on teaching materials and cohort policies. Currently our R resources are all in Jupyter Notebook (.ipynb) format, but Jupyter isn't always the best way to use R. The app, RStudio, is an Integrated Development Environment (IDE) for R and is generally a smoother experience. It doesn't support Jupyter Notebook files, however.

Talk to your CSM to find out what your cohort's employer(s) will support.

Option 1 - The 'pure' R way

1. Install R from CRAN (Comprehensive R Archive Network) - https://cran.r-project.org/
2. Install RStudio Desktop from RStudio - https://www.rstudio.com/products/rstudio/download/
   Warning: there is a version of RStudio available within Anaconda. Do not use this, get RStudio from the source.
3. Teach R using .R scripts or RMarkdown (.Rmd) files, copying from the Module 3 Jupyter Notebook

Option 2 - Keep it simple, Anaconda and Jupyter only

1. Install R through Anaconda (instructions below)
2. Use Jupyter to teach using the Module 3 Jupyter Notebook

Option 4 - Use RStudio Cloud - no installation required

But you'll need to copy the Jupyter Notebook into an .R script or RMarkdown file. Check in advance of the session that your apprentices' firewalls or security policies don't block them from accessing RStudio Cloud

Option 5 - Use another cloud-based option for Jupyter Notebooks.

You might be using Google Collab, Deepnote, or similar to run your Python bootcamps. Many of these tools support R as well. Make sure you check that the notebook runs successfully in whatever online tool you use well in advance of the session!

RStudio:

This is the easier case, if your apprentices can get RStudio. Run through the following steps with them in advance of your R session to check that they can install the libraries they need.

Step 1:

In RStudio, run the code `install.packages('tidyverse')`
This will download and install the code libraries you will need for the session. This will usually take several minutes to run, so you might want to get apprentices to start it just before a break in your delivery session. If the apprentices' firewall blocks them from accessing the URLs where the code is hosted, they'll get an error message at this step. To resolve this, raise it with the CSM - they'll need to talk to the apprentices' company's IT team to get the packages installed.
Alternatively, you could have the apprentices speak to their IT team (or managers) themselves, if you think they might be able to request the installation themselves.

Step 2:

Once the `install.packages('tidyverse')` code has run (or the tidyverse library has been installed by some other route) apprentices should run `library(tidyverse)` inside RStudio and check they get no error messages.

If these two steps are completed, your apprentices are ready for their R session in RStudio!

R through Anaconda:

This is the harder case, but can work fine with some care. You have to be quite particular about how you use R in Anaconda. Guide your apprentices through the steps below to make sure they have a working R installation

Step 1:

Open Anaconda Prompt (searching in your start menu is probably the easiest way to do this. Within Anaconda Prompt run the following commands, in order (the first one will take around 10 minutes to complete, so maybe get apprentices to start it just before a break):

```
conda create -y -n r_env r-essentials jupyterlab
```

```
conda activate r_env

jupyter notebook
```

This should open Jupyter notebooks. (Note: if you don't use JupyterLab, but just regular Jupyter, then you can remove `jupyterlab` from the first command, and this might speed up the installation slightly).

If the first step of this fails, get screenshots of any error messages and check the df-team Slack channel for help, but it's probably a firewall issue and will need CSM support.

NOTE:

You have just created an R environment (named r_env) within Anaconda. Every time you want to use R through Anaconda, you will need to first activate that environment by opening Anaconda prompt and running the command

`conda activate r_env`

You can then launch Jupyter notebooks from the Anaconda prompt as above and have R available. You can return to your base environment by running

`conda deactivate`

although anytime you quit Anaconda and re-open it should revert to your base environment by default. In Anaconda prompt, at the left of the screen should be the name of the active environment in parentheses. So when you open Anaconda prompt, it should say `(base)`, and after activating your r_env environment it should instead say `(r_env)`

Step 2:

Within Jupyter (and within your R environment, which should be active after Step 1), go to the "New" dropdown menu at the top right, and under "notebook" select "R". This should open a blank notebook, and in the top right you should see the R symbol instead of the Python symbol.

Within a cell of the notebook, run the command `library(tidyverse)`

You should get a message (screenshot below) , which looks like an error, but isn't! If this code runs without actual errors (which will helpfully start with the word "error!"), you're good to go - your apprentices can use R and all the packages they need.

```
library(tidyverse)
```

```
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Registered S3 method overwritten by 'rvest':
  method            from
  read_xml.response xml2
-- Attaching packages --------------------------------------- tidyverse 1.2.1 --
<U+221A> ggplot2 3.1.1       <U+221A> purrr   0.3.2
<U+221A> tibble  2.1.1       <U+221A> dplyr   0.8.0.1
<U+221A> tidyr   0.8.3       <U+221A> stringr 1.4.0
<U+221A> readr   1.3.1       <U+221A> forcats 0.4.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Fig 1. Attach messages from `library(tidyverse)` in Jupyter. This is **not** an error, it's just telling you that some commands from the library you imported overwrite old commands and that the tidyverse core packages were loaded correctly.

```
data$x
```

```
Error in data$x: object of type 'closure' is not subsettable
Traceback:
```

Fig 2. This **is** an error. If you see something like this, and aren't confident to fix it yourself, screenshot the error message and ask others in the DF team for help - this is why you're testing the setup well in advance of the delivery session!

WARNING:

Your apprentices will need to run `conda activate r_env` every time before launching a Jupyter notebook to have access to R.

Be warned that the notebook included in the session materials contains the code `install.packages('tidyverse')` but you should **NEVER** run this code in R within Anaconda. Mixing R's inbuilt `install.packages()` function with Anaconda package installation is a recipe for disaster.

You installed tidyverse when creating the environment, you don't need to install it again within R - though you will need to run `library(tidyverse)` in any R notebook where you want to use functions from tidyverse. This is just like calling `import <package name>` in Python.

If you (or your apprentices) ever need other R libraries in future, you will need to create a new R environment with those libraries in. The code (in Anaconda prompt) is:

```
conda create -n <env-name> r-essentials <package-name>
```

where `<env-name>` should be replaced with whatever you want to call your environment (we used r_env above) and `<package-name>` should be replaced with the name of the package you want to install, prefixed with "r-". You can list multiple packages separated by spaces. For instance, to install the dplyr and stringr packages in an environment, you would run

```
conda create -n <env-name> r-essentials r-dplyr r-stringr
```

This command will ask for confirmation to proceed partway through. You can skip this step by adding `-y` to the command as we did in Step 1. (Note: dplyr and stringr are both included in the r-essentials package, so you wouldn't actually need to install these separately - this is just for illustration).

Note also that some R packages are not available through Anaconda. If you find yourself needing plenty of packages you can't get through Anaconda, it's probably best to ditch Anaconda and go to the classic R approach (see option 1 in the Preparation section!). If you do want to mix and match packages from Anaconda and CRAN, the "r-" prefix only applies to Anaconda versions of the packages - so "r-dplyr" is just called "dplyr" everywhere outside Anaconda, for example.