Keeping an exotic pet in your home!

Taming Python to live in RStudio because sometimes the best language is both!

Emma Rand
emma.rand@york.ac.uk

Materials: https://github.com/3mmaRand/useR2019_tutorial

But first.....Who am I?
and some Thank yous!

Materials: https://github.com/3mmaRand/useR2019_tutorial

Who I am?

Lecturer (Assistant Professor), Department of Biology University of York, UK

I'm a biologist by training.

Long time R user, relatively new to Python.

[←] York, twinned with Dijon.

Materials:
https://github.com/3mmaRand/useR2019_tutorial

Thank yous!

To the local organisation committee

- Nathalie Vialaneix (chair), MIAT, INRA
- Sébastien Déjean (vice-chair), Institut de Mathématiques de Toulouse, Université Toulouse 3 Paul Sabatier
- Anne Ruiz-Gazen (vice-chair), Toulouse School of Economics, Université Toulouse 1 Capitole
- Heather Turner (vice-chair), statistical consultant and associate fellow of the Statistics Department at the University of Warwick
- Aurore Archimbaud, Toulouse School of Economics
- Christophe Bontemps, Toulouse School of Economics, INRA
- Robert Faivre, MIAT, INRA
- Xavier Gendre, Institut de Mathématiques de Toulouse, Université Toulouse 3 Paul Sabatier
- Thibault Laurent, Toulouse School of Economics, CNRS
- Élise Maigné, Observatoire du Développement Rural, INRA
- Pierre Neuvial, Institut de Mathématiques de Toulouse, CNRS
- Rémi Servien, InTheRes, INRA
- Matthias Zytnicki, MIAT, INRA

Materials: https://github.com/3mmaRand/useR2019_tutorial

Thank yous!

For funding

To the organisers and their sponsers

My institution

Materials: https://github.com/3mmaRand/useR2019_tutorial

Thank yous!

Leila Khajavi

Leila is an American pursuing her PhD in Bioinformatics here in Toulouse, affiliated with both MIAT (INRA) and CPTP (INSERM).

She is very kindly giving her time here today to help out and has already contributed to the session by going through the material and giving some feedback.

But any errors that remain are mine!

Materials: https://github.com/3mmaRand/useR2019_tutorial

Thank yous!

My colleagues at the University of York allowed me to practice on them:
James Chong
Bryden Fields
Martina Stoycheva
Jack Law
Oliver Noble
Rebecca Hall
Evie Farnham
Mike Gray

And finally...

Everyone here for coming! 

Materials: https://github.com/3mmaRand/useR2019_tutorial

Who are you?

- ○ an undergraduate student
- ○ a postgraduate student
- ○ an early career researcher/academic
- ○ an established career researcher/academic
- ○ an early career 'data' professional
- ○ an established career 'data' professional

Page 1 sur 4

SUIVANT

N'envoyez jamais de mots de passe via Google Forms.

Why `reticulate`?

You finally found the **perfect** solution to a data problem!

😁

but it's written in Python

😬

You're collaborating with some great people

😁

but they mainly use Python!

😬

You want to use existing/available solutions and collaborate more easily.

Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

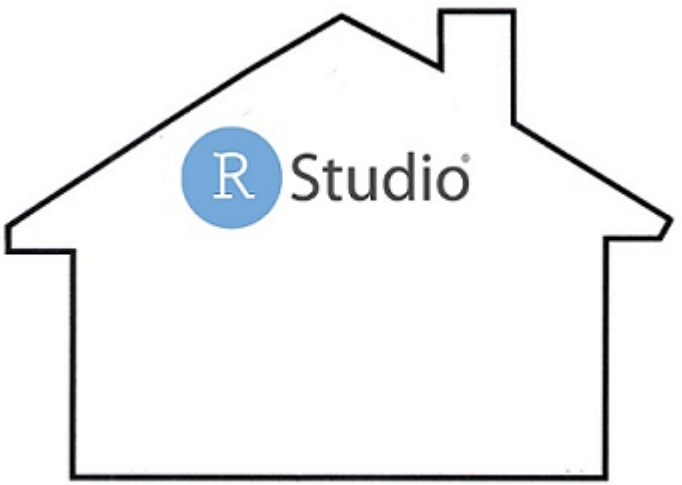
You could move to the darkside 🤕...

But you're familiar 💗 with R...

Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

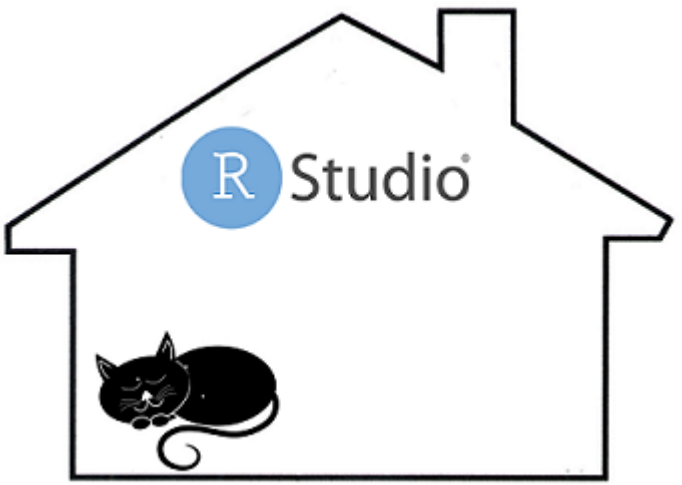
But you're familiar with R
and very at home in RStudio...



Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

But you're familiar with R
and very at home in RStudio
because it's comfortable...



Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

But you're familiar with R
and very at home in RStudio
because it's comfortable
and has many tools you like...

Materials: https://github.com/3mmaRand/useR2019_tutorial

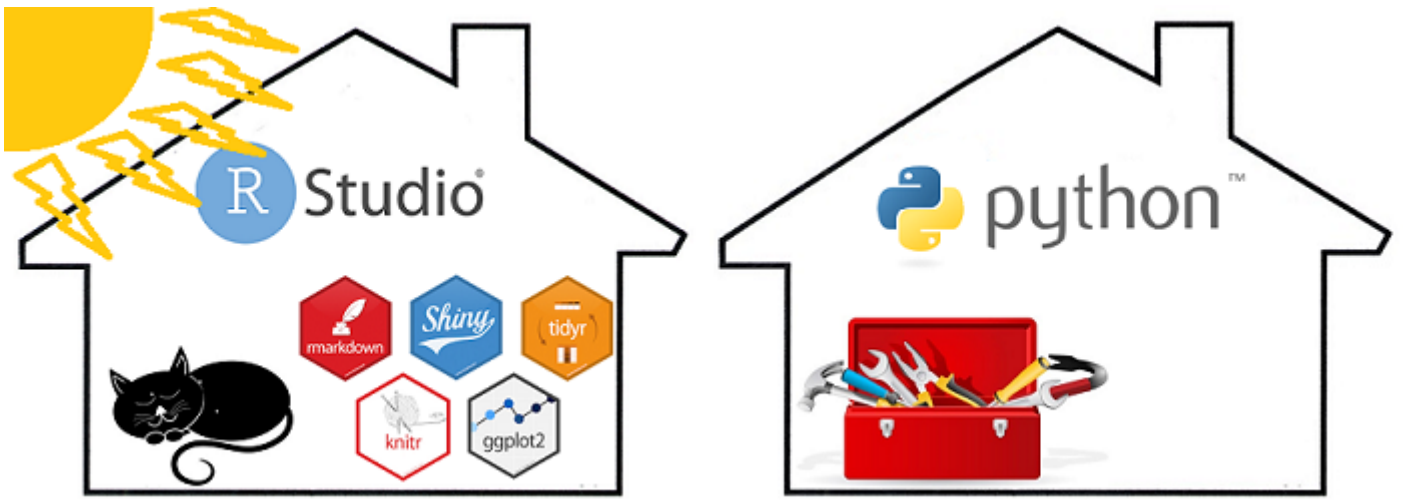
Why `reticulate`?

But you're familiar with R
and very at home in RStudio
because it's comfortable
and has many tools you like
and the sun always shines!

Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

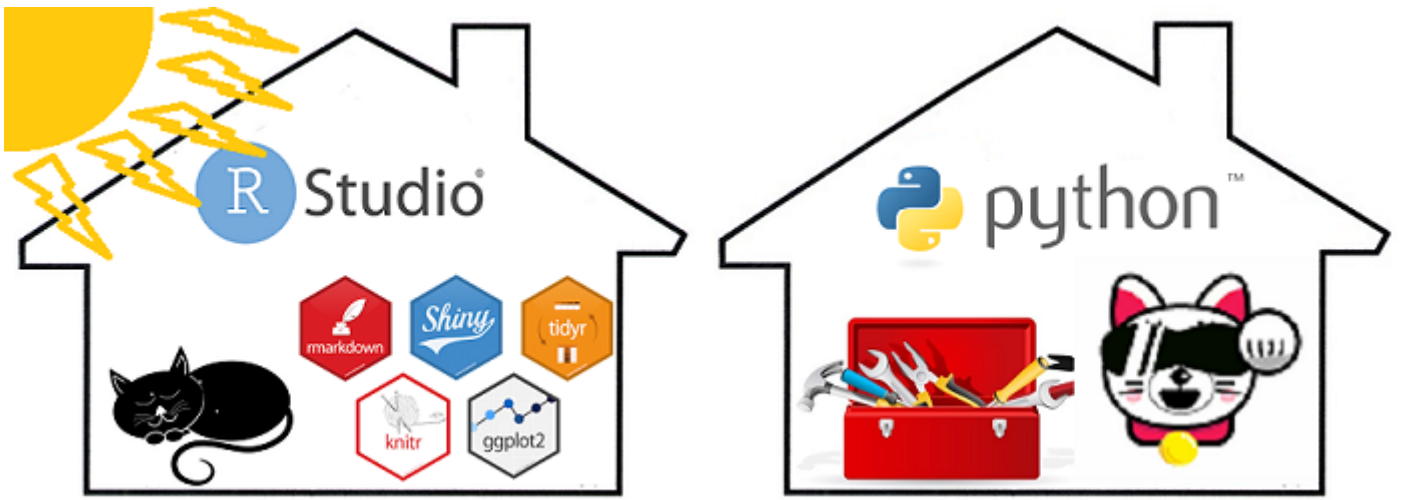
So even though Python has some great tools...



Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

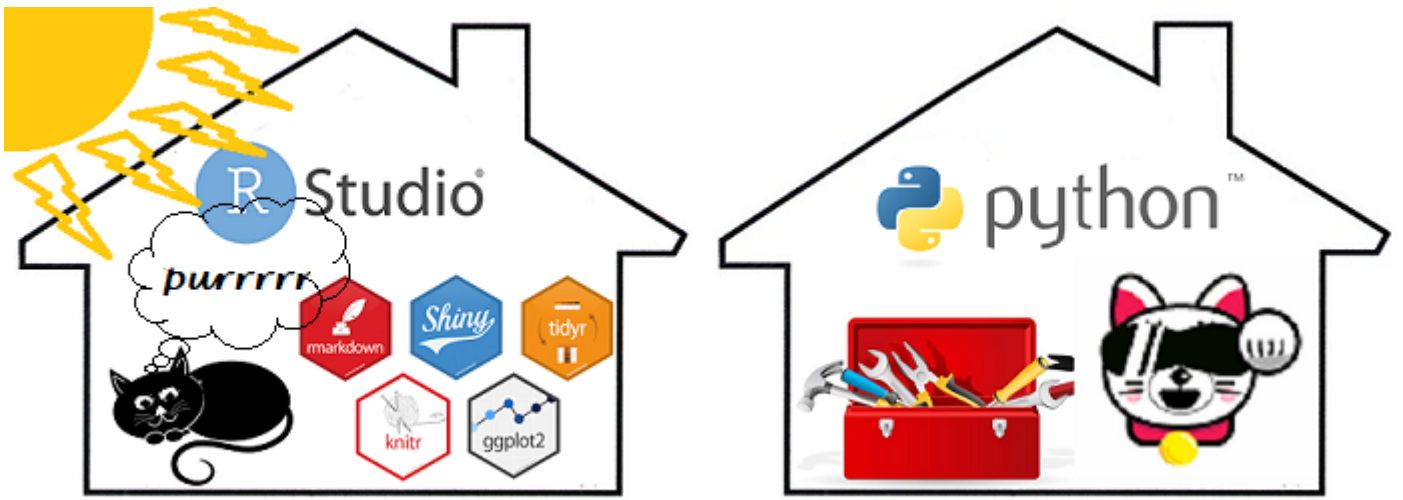
So even though Python has some great tools
and some very cool people...



Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

So even though Python has some great tools
and some very cool people
you don't want to move in!

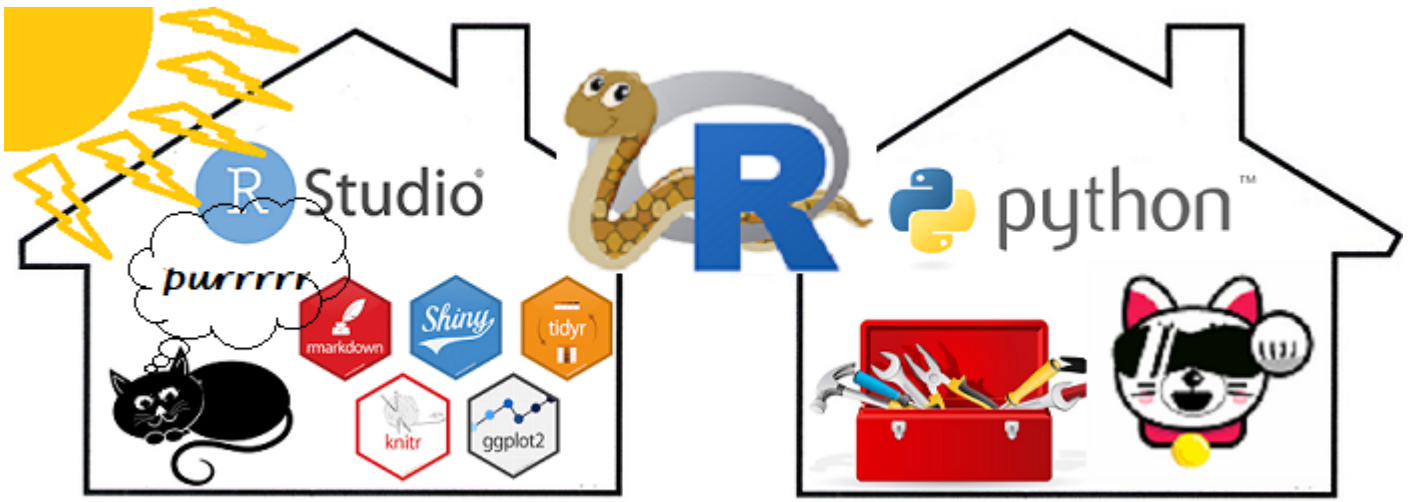


Materials: https://github.com/3mmaRand/useR2019_tutorial

Why `reticulate`?

Thankfully, there's `Reticulate`

JJ Allaire, Kevin Ushey and Yuan Tang (2018). reticulate: Interface to 'Python'. R package version 1.10. https://CRAN.R-project.org/package=reticulate



Materials: https://github.com/3mmaRand/useR2019_tutorial

Tutorial Overview

Tutorial Overview 

- Context: Very brief background to R 💜 and Python 🤝
- Rationale: why use Reticulate, what are its key features, how can it be used and set up
- Background: A little Python. Only what you need to start making sense of calling Python from R
- Using Python interactively with `repl_python()`
- Integration in R markdown 🖋: Writing your own `reticulate` tutorial
    - Part 1: Building your understanding
    - Part 2: Classifying audio segments

What the tutorial won't be: a thorough introduction to Python, R Markdown or machine learning.

Materials: https://github.com/3mmaRand/useR2019_tutorial

useR2019 ReticulateTutorial: Participants log

Let me know what you're experiencing!

Use smileys, single words or sentences. The time stamped responses will help me understand how to develop the tutorial.

Votre réponse

ENVOYER

N'envoyez jamais de mots de passe via Google Forms.

Aims

This tutorial was designed for:

- beginner to intermediate R users
- those with little to no experience of Python
- those who may not have experience of Rmarkdown

By the end of the tutorial you should be able to:

- pass objects between simultaneous R and Python sessions
- use Python methods in R code
- incorporate Python snippets in to your R workflow even if you don't understand in detail how they work

As long as you know what the Python methods and code snippets are for, and have a good-enough understanding of their inputs and outputs, then you can use `reticulate`.

Some of the code is specifically designed to develop understanding of the integration rather than reflect a workflow. The later section comprises an example workflow.

Background

	R	Python
Released	1995	1991
Author	Ihaka & Gentleman, Chambers	van Rossum
Purpose	User-friendly data analysis and visualisation for 'non-programmers'	Object-oriented, Readable, general purpose programming language
Users most likely to be	Statistics graduates, Academics, data scientists	Computer Science graduates, Software engineers
Features	R Markdown, tidyverse	Integration with web aps, Unified Machine learning API

Materials: https://github.com/3mmaRand/useR2019_tutorial

Why use Reticulate

Speed up your workflow

- Problem solving is the defining feature of a data scientist
- Language should be secondary
- Choice of language driven by early impressions of the data. Change in direction later means lost time in translating

Facilitate collaboration

- Allows you to leverage the skills expertise of the whole team
- Solves the hardest problem in Data Science - People[1].
- Many Data Scientists know both and they are happier[2]

[1] Mangano, 2019
[2] Stack Overflow Developers' Survey, NanoMathius, 2018

Reticulate

Key features

1. Ability to call Python from R
2. Translation between R objects and Python objects
3. Flexible binding to different Python environments

Reticulate

Alllows you to use Python in four ways:

1. Interactively in the console: `repl_python()`
2. Sourcing Python scripts
3. Importing Python modules
4. In R Markdown documents

We will start with `repl_python()` to build our understanding.

Then use R Markdown.

Reticulate

Ingredients

You will need

- RStudio 1.2
  1.2 is needed for some of the most useful features
- Python
  Anaconda 3 recommended for data science, includes many useful libraries.
- The `reticulate` package. I recommend using the development version

```
devtools::install_github("rstudio/reticulate")
```

This one

Not this one

Reticulate

You will also need

- Any other Python modules your Python code depends on (not needed here)
- Probably / possibly.....to set to the QT_PLUGIN_PATH environment variable.

  In windows: Control Panel -> System and Security -> System then

  Advanced System settings -> Environment variables

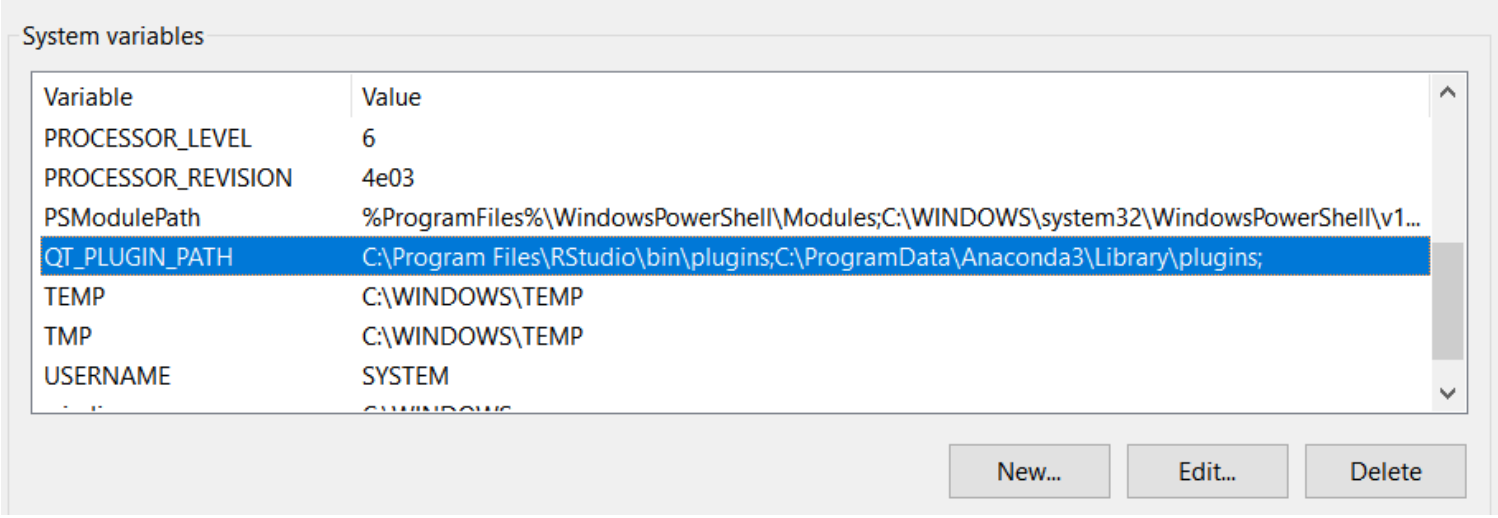
  I have set mine to:

```
C:\Program Files\RStudio\bin\plugins;
C:\ProgramData\Anaconda3\Library\plugins
```

If you can describe and resolve in a better but still minimal way, please get in touch!

Reticulate

In windows: QT_PLUGIN_PATH environment variable

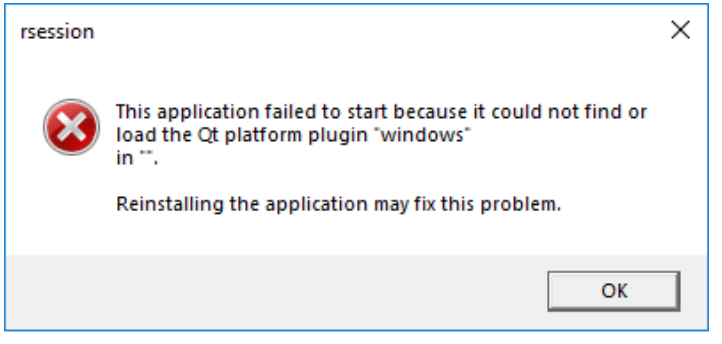


Reticulate

In windows: QT_PLUGIN_PATH environment variable

If you get this error:

This application has failed to start because it could not find or load the qt platform plugin "windows" in ""



Setting the QT_PLUGIN_PATH environment variable as on the previous slide should fix it.

If you can describe and resolve in a better but still minimal way, please get in touch!

A little Python

Python fundamentals

Suppose you wanted to created an array of 5 numbers.

In R you might do this as:

```
r_array <- c(4, 5, 1, 6, 8)
```

In Python you might use a list[1]

A list is created like this:

```
python_list = [4, 5, 1, 6, 8]
```

Python uses = for assignment

The square brackets denote a list

[1] Python doesn't have a native array data structure

Python fundamentals

But lists do not behave as a R user might expect.

For example, what would you expect the output to be?

```
python_list = [4, 5, 1, 6, 8]
python_list * 2
```

This?

```
python_list = [4, 5, 1, 6, 8]
python_list * 2
[8, 10, 2, 12, 16]
```

Infact it is this:

```
python_list = [4, 5, 1, 6, 8]
python_list * 2
[4, 5, 1, 6, 8, 4, 5, 1, 6, 8]
```

😳

Python fundamentals

Instead you might use the NumPy package[1]. NumPy arrays behave like R vectors/arrays.

To make a NumPy array we need to first `import` NumPy, then use it's `array()` function.

This is going to introduce us to several Pythonesque things.

[1] NumPy is the fundamental package for scientific computing with Python. It is part of the SciPy ecosystem.
Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.

Python fundamentals

The Python code looks like this:

```
import numpy as np
python_array = np.array([4, 5, 1, 6, 8])
```

Python fundamentals

The Python code looks like this:

```
import numpy as np
python_array = np.array([4, 5, 1, 6, 8])
```

`import` in Python is the equivalent of `library()` in R

To use methods in NumPy (and other modules) we need to use the "dot" notation:

`numpy.method_name()`

To make this quicker to type it is common to use an alias. That's the `as np` bit

Python fundamentals

```
import numpy as np
python_array = np.array([4, 5, 1, 6, 8])
```

The second line of code creates the numpy array (from a list).

To do things with `python_array` we might use a built-in function. These are used in a way that will be familar to you, for example:

```
type(python_array)
<class 'numpy.ndarray'>
```

Python also has methods. Methods are called on objects with the dot notation.
For example:

```
python_array.max()
8
```

Enough!

Let me code

We will cover some more Python as we go through the tutorial.

Using Python in the console

```
reticulate::repl_python()
```

Using Python in the console

Steps

We are going to use Python interactively in the console.

We will

- Create a new project
- Check our Python environment
- Start a Python session from our R session
- Create a NumPy array
- Use the NumPy array
  - find its size (an attribute)
  - calculate its mean (a method)
- End the Python session
- Access the NumPy array from our R session

Extra exercise Indicates an optional extra exercise to try while you're waiting for me to move on.

Using Python in the console

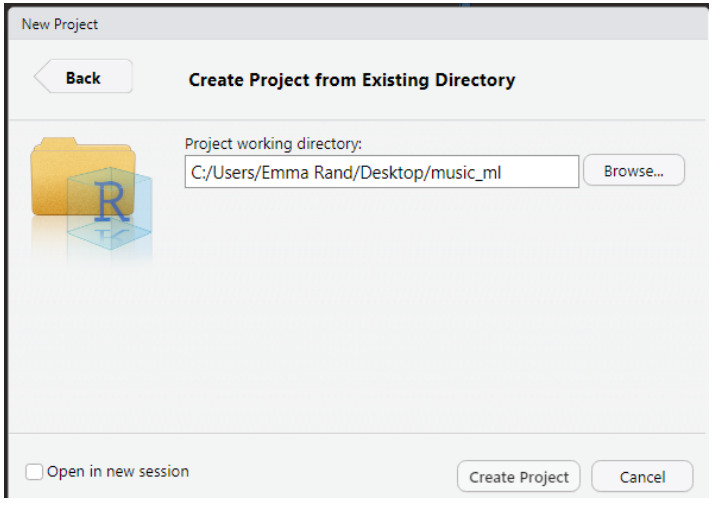
Organising ourselves

We are going to work in a project[1].

File | New Project | Existing Directory

Choose Browse

And navigate to the "music_ml" folder and Open



Choose Create Project

[1] New to projects? RStudio: Using Projects

Using Python in the console

We are going to be moving between Python and R sessions.

Which could get confusing!

What session you are in **at the start of a slide** is indicated like this:

Sub-heading

You are working interactively with python and your prompt should look like this:

```
>>>
```

You may **finish** the slide in an R session

Sub-heading

You are working interactively with R and your prompt should look like this:

```
>
```

You may **finish** the slide in a Python session

Using Python in the console

Prepare to start a Python session

Load the reticulate package

```
> library(reticulate)
```

For this section on using `repl_python()` I will show the prompt, either `>` or`>>>` in the code.

Using Python in the console

Start a Python session

With `repl_python()`:

```
>  repl_python()
Python 3.7.3 (C:\PROGRA~3\ANACON~1\python.exe)
Reticulate 1.12.0.9003 REPL -- A Python interpreter in R
>>>
```

Using Python in the console

repl_python()

```
>  repl_python()
Python 3.7.3 (C:\PROGRA~3\ANACON~1\python.exe)
Reticulate 1.12.0.9003 REPL -- A Python interpreter in R
>>>
```

- You get a message to tell you what version of Python you're using and where it is

Using Python in the console

repl_python()

```
>  repl_python()
Python 3.7.3 (C:\PROGRA~3\ANACON~1\python.exe)
Reticulate 1.12.0.9003 REPL -- A Python interpreter in R
>>>
```

- You get a message to tell you what version of Python you're using and where it is.
- `>>>` indicates the Python prompt.

We need to use Python 3.

By default, `reticulate` uses the version of Python found on your PATH.

If Python 3 is not being used (possibly Mac users) we can change it.

Using Python in the console

Change the Python version.

It is not necessary to do this if you're already using Python 3 but it won't hurt if you do.

End the `repl_python` session:

```
>>> exit
>
```

Set the version of Python you want to use.

This needs to be where Anaconda3 installed. In my case:

```
> use_python("C:/ProgramData/Anaconda3/python.exe")
```

Using Python in the console

Check the Python version

Confirm it has been set:

```
> Sys.which("python")
                                    python
"C:\\PROGRA~3\\ANACON~1\\python.exe"
```

On windows machines the paths will be short paths (8+3 components, no spaces) with \ as the path delimiter.

In my case, it is short for `C:/ProgramData/Anaconda3/python.exe`

So I know I'm using Python 3

With `repl_python()`:

```
>  repl_python()
Python 3.7.3 (C:\PROGRA~3\ANACON~1\python.exe)
Reticulate 1.12.0.9003 REPL -- A Python interpreter in R
>>>
```

Using Python in the console

Make a NumPy array

Import the NumPy package

```
>>> import numpy as np
>>>
```

You get command completion!

and create an array in the console

```
>>> python_array = np.array([4, 5, 1, 6, 8])
>>>
```

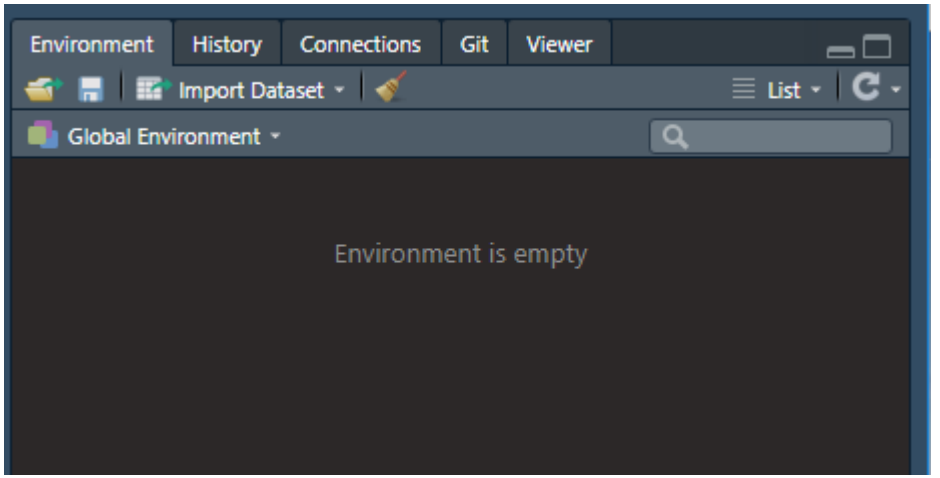
Success!

Extra exercise 1. Create a list called `python_list`
Extra exercise 2. Import pandas and create a small data frame. You'll need the DataFrame() method from pandas. As arguments, it will take lists of lists the same length

Using Python in the console

Oh! 😕

But nothing appears in your R environment!



Extra exercise 1. Create a list called `python_list`
Extra exercise 2. Import pandas and create a small data frame. You'll need the DataFrame() method from pandas. As arguments, it will take lists of lists the same length

Using Python in the console

Where is our NumPy array?

Can Python can find it?

🔎

```
>>> python_array
array([4, 5, 1, 6, 8])
>>>
```

👍 🎉

Great, the Python session can see it!

Extra exercise 3. How do you need to organise the list of lists in the DataFrame() command to get a 4 rows and 2 columns compared to 2 rows and 4 columns.

Using Python in the console

Work with a NumPy array

Check the object type of `python_array`:

```
>>> type(python_array)
<class 'numpy.ndarray'>
>>>
```

Find the size of `python_array` with:

```
>>> python_array.size
5
>>>
```

`size` is an *attribute* or *value* of NumPy arrays.

Extra exercise 4. What does `.size` give for the the list and dataframe? Try `.shape`

Using Python in the console

Work with a NumPy array

Can you find the mean of `python_array`?

```
>>> python_array.mean()
4.8
>>>
```

`mean()` is an *method* of NumPy arrays.

Extra exercise 5. Find other summary statistics for the array
Extra exercise 6. Is there a mean method for the dataframe? What about the list? Can you find the mean of your list?

Using Python in the console

Bye bye Python

Let's exit Python

```
>>> exit
>
```

Using Python in the console

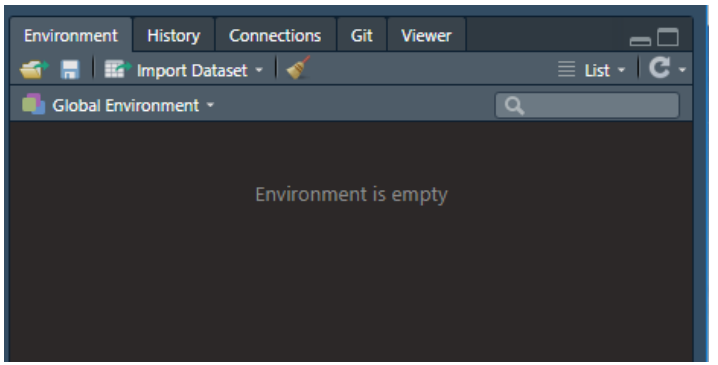
Hello R

```
>>> exit
>
```

The R prompt is returned

Using Python in the console

Now where is `python_array`?



Our environment is still empty

But we **can** access the Python object using `py$...`

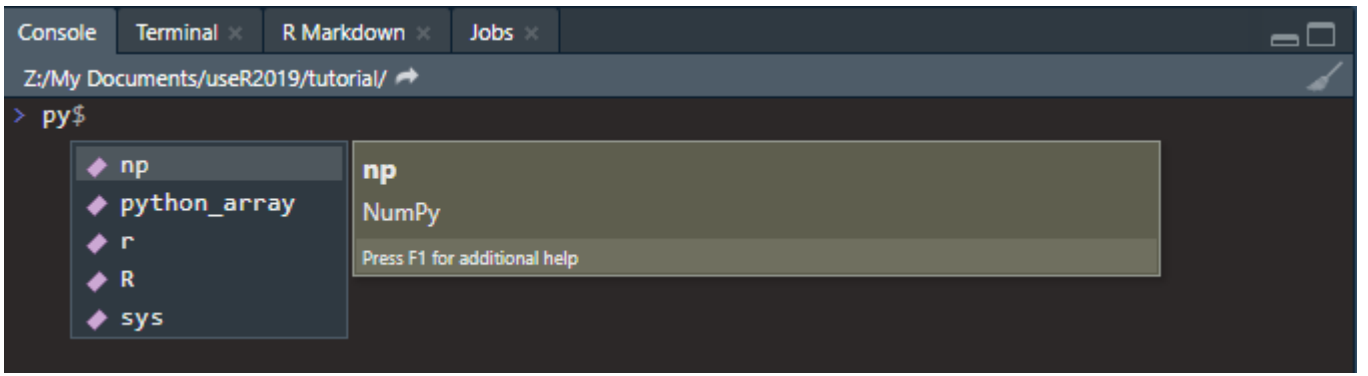
```
> py$python_array
[1] 4 5 1 6 8
```

Extra exercise 7. Can you access the list and dataframe?

Using Python in the console

Accessing python-created objects

And we have normal command completion so typing `py$` followed by the TAB key ↹ will list the available python objects.



Notice we have more than `python_array`.

We will return to this in a few slides.

Using Python in the console

Accessing python-created objects

`reticulate` makes these python-created objects behave how you expect them to in R.

```
> py$python_array * 2
[1]  8 10  2 12 16
```

```
> py$python_array * py$python_array
[1] 16 25  1 36 64
```

Extra exercise 8. How do the list and dataframe behave?

Using Python in the console

R functions on python-created objects

They behave the way you expect them to because they are converted R objects when used!

```
> class(py$python_array)
[1] "array"
```

`py$python_array` is a one-dimensional array

numpy.ndarray $\longrightarrow$ array

And look! 👀 We can use R functions on python-created objects

Extra exercise 9. What is the object translation for the list and dataframe?

Using Python in the console

R functions on python-created objects

The mean...

```
> mean(py$python_array)
[1] 4.8
```

... and length of the array

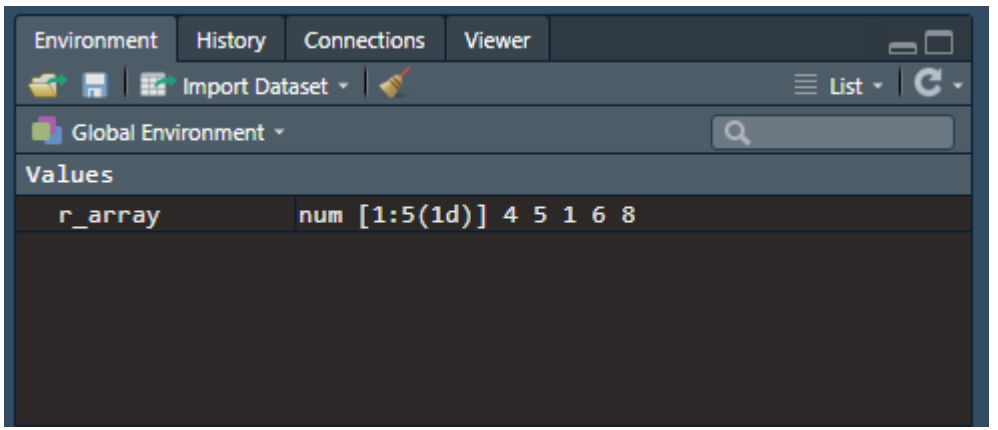
```
> length(py$python_array)
[1] 5
```

Using Python in the console

R functions on python-created objects

You can assign the python object explicitly

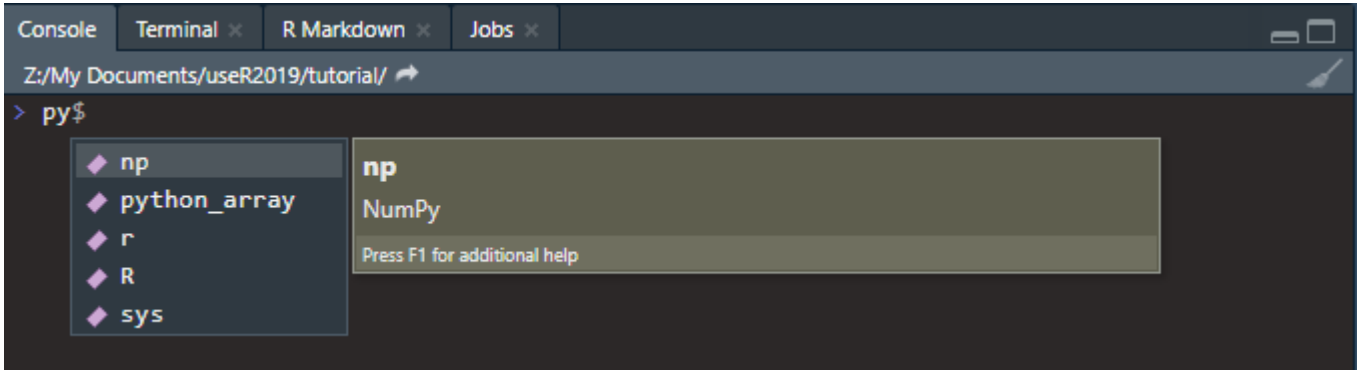
```
> r_array <- py$python_array
```



Using Python in the console

Python methods on python-created objects

Remember this?



We have access to the `np` object which results from importing `NumPy` as `np` in our Python session.

This gives us access to all the NumPy methods.

Using Python in the console

Python methods on python-created objects

In Python `np` methods are applied by following the object with a dot:

`object_name.method_name()`

For example, we found the mean of `python_array` with:

```
>>> python_array.mean()
4.8
```

In R we access and apply `np` methods in a R like way:

`py$np$` *`method_name(object_name)`*

So to use the mean method on `py$python_array`:

```
> py$np$mean(py$python_array)
[1] 4.8
```

This would be a strange thing to do!

Extra exercise 10. Create a vector, array and/or dataframe in R then restart `repl_python()` and access them in the Python session with `r.`

Using Python in the console

Why strange??

Because if you already have a python-created object on which you wanted to use some Python methods before doing further work in R, you would more naturally

- Use those methods in Python in a Python way
- Then access the result in R for further work

Than

- access the object in R
- apply python methods to a python object in an R-like way in an R session
- Then do your further work

Extra exercise 11. In your `repl_python` session, apply R functions to the data structures made in the R session

How: Using Python in R Markdown

How: Using Python in R Markdown

Aim

You are going to write your own `reticulate` tutorial by combining your own notes and investigations with the code given in the slides.

The tutorial will be in two parts:

- The first part of the tutorial aims to develop our baseline understanding of the link between the R and Python sessions and running code chunks interactively. We will pass simple objects between sessions more than you would normally just for demonstration purposes

- The second part covers the importing, modelling and visualisation of processed audio data.

The tutorial is "Classification of Audio segments by instrument: A Tutorial on using the R package reticulate to integrate R and Python."

You are aiming for something like this: Classifiying_music.html

How: Using Python in R Markdown

The data

We are going to work with some data derived from 9 pieces of music. Three examples are:

- Chopin - Ballade No. 1 in G Minor
- Corelli - Sonata da Chiesa, Op. 1 No. 1 in F major
- Mozart - Sonata in F major for piano and violin K 376

The example and the original Python code to process the audio files and carry out the machine learning analysis methods are by Michael Knight, University of Bristol.

Each of the pieces of music has been segmented into 5-second segments each of which has 5000 features. The features represent the apodised power spectrum of a 5-second segment.

We will try to classify these segments.

How: Using Python in R Markdown

The data

Up to 100 segments were taken from each audio file although there are fewer for pieces shorter than 500 seconds.

Instrument	Piece	Number of segments
Piano	01 Ballade No. 1, Op. 23.m4a	100
Piano	1-01 Sonata No. 1 In F Sharp Minor, Op. 11_ I. Introduzione. un Poco Adagio ,Allegro Vivace.m4a	100
Piano	5-01 Beethoven_ Piano Sonata No. 14 in C-Sharp Minor, Op. 27 No. 2, 'Moonlight'_ I. Adagio sostenuto.m4a	74
Piano	9-05 Beethoven_ Piano Sonata No. 29 in B-Flat Major, Op. 106, 'Hammerklavier'_ I. Allegro.m4a	100
Violin	05 No.2 in A major RV 31_ I. Preludio a Capriccio. Presto.m4a	14
Violin	1-01 Sonata da chiesa a tre in F Major, Op. 1, No. 1_ I. Grave.m4a	15
Violin	1-02 Sonata da chiesa a tre in F Major, Op. 1, No. 1_ II. Allegro.m4a	18
Violin	24 No.11 in D major RV 9_ II. Fantasia. Presto.m4a	16
Violin and Piano	3-01 Sonata for Piano and Violin in F, K. 376_ I. Allegro.m4a	59

How: Using Python in R Markdown

The data

There are two xlsx files for each piece.

- *name*_segments.xlsx
  has the segments in rows and the features in columns
- *name*_SegmentInfo.xlsx
  has the metadata for each segment: the name of the piece, the instrument label on the piece, the start and end time of the segment.

There are 496 segments in total of which 374 are from piano pieces and 63 are from violin pieces. The remaining 59 pieces are from Mozart's Sonata piano and violin.

How: Using Python in R Markdown

What is R Markdown

Live demo

May not be needed depending on the audience responses from earlier.

Just watch for a while....

How: Using Python in R Markdown

What is R Markdown

Key points from the demo

- blends narrative text with analysis code and output
- human readable
- YAML header between the ---
- code chunk options control whether the code and its output end up in your 'knitted' document
- comments
  - in a code chunk the # is still used for comments
  - in the text a comment is written like this <!-- a comment -->
  - but use Ctrl+Shift+C
- # in the text indicate headings

How: Using Python in R Markdown

Make your own R markdown doc

File | New File | R Markdown

Add a title.

You could use your own or copy and paste a title from Classifiying_music.html

Add your name

Delete everything except:

- the YAML header between the ---
- the first code chunk which begins:
  ```{r setup, include=FALSE}
  ```

How: Using Python in R Markdown

Set up your default code chunk behaviour

That first code chunk is for setting some **default** code chunk options.

I often use these:

````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE,
                      warning = FALSE,
                      message = FALSE)
```
````

`echo = FALSE` means the code will not be included by default - this is normally what you want in a report.

However, I used `echo = TRUE` in Classifiying_music.Rmd so you could see the code.

Any output is included by default

How: Using Python in R Markdown

Start adding some text

Add a little introduction.

You can make your own notes or copy and paste from Classifiying_music.html

Save your file The Save directory will be the Project directory, "music_ml". Do not change that!

How: Using Python in R Markdown

Add an R chunk

The second code chunk in an Rmd document typically loads the required packages.

There's one package you will definitely need: `reticulate`

We also also use `readxl`[1] and `ggplot2`[2] so let's add those too:

````
```{r pkgs}
##########   R    ##########
library(reticulate)
library(readxl)
library(ggplot2)
```
````

Use Insert | R to add a code chunk. From now on

Insert R chunk Insert Python chunk

[1] Wickham, H. Bryan, J. (2019). readxl: Read Excel Files. R package version 1.3.1. https://CRAN.R-project.org/package=readxl
[2] Wickham, H. (2016) ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York https://ggplot2.tidyverse.org

How: Using Python in R Markdown

Run a R chunk!

To run R commands interactively in a R Markdown document we need to run this code chunk.

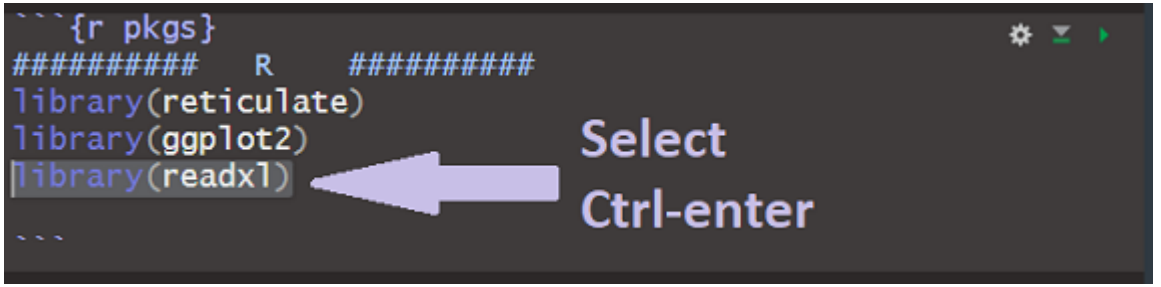
Use the green arrow on the right of the chunk to run it.

From now on
=> Run Chunk
means use the green button to run the chunk.

How: Using Python in R Markdown

Run a line in a chunk

Sometimes we will modify a chunk. In this case, you don't need to run the whole chunk, just the added line. You can do this by selected the line and doing Ctrl-enter[1]



From now on
=> Run added line
means you don't need to run the whole chunk, just the most recently added line.

[1] Unless you have set the short cut to something else.

How: Using Python in R Markdown

Add a Python chunk

Now we'll add a chunk to import the Python modules we will need.

In Part 1, we will use the `os` module which provides a way of using operating system dependent functionality.

Insert Python chunk **=> Run Chunk**

```
```{python mods}
########## PYTHON ##########
import os
```
```

Feeling unnerved by not being able to see that anything has happened?

You can check the Python session on the command line in the console: `py$` should give you command completion options which include `py$os`

We will add any other Python modules we need later.

How: Using Python in R Markdown

Part 1 Building our understanding

If you get the error:

This application has failed to start because it could not find or load the qt platform plugin "windows" in ""

You may need to check QT_PLUGIN_PATH (see earlier slide)

How: Using Python in R Markdown

Part 1 Building our understanding

You may want to add a header indicating this is Part 1 of the tutorial

A Python equivalent for R's `getwd()` is `getcwd()` from the `os` module.

If you have successfully imported the `os` module you can use it like this:

Insert Python chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
os.getcwd()
```
````

```
 'C:\\Users\\Emma Rand\\Desktop\\useR2019\\useR2019_tutorial\\music_ml'
```

It prints out!

On a windows machine the 'windows-way' round slashes are escaped
On a mac the path will be given 'correctly'

How: Using Python in R Markdown

Part 1 Building our understanding

We can also use the the `getcwd()` method from `os` in a R chunk.

Remember, to access Python objects (of any sort) in R we use `py$`

Here we access the `os` object's methods

BUT in a R-like way using the `$` not a `.`

Insert R chunk **=> Run Chunk**

```
```{r}
##########   R    ##########
py$os$getcwd()

```
```

```
 [1] "C:\\Users\\Emma Rand\\Desktop\\useR2019\\useR2019_tutorial\\music_ml"
```

How: Using Python in R Markdown

Part 1 Building our understanding

And we access R functions in Python chunks in a Python-like way

Can you do that?

Insert Python chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
r.getwd()
```
````

```
 'C:/Users/Emma Rand/Desktop/useR2019/useR2019_tutorial/music_ml'
```

That's right! Though I'm not sure quite why you would!

👀 You might have noticed that the output of Python chunks doesn't include the element numbering [1] characteristic of R output.

How: Using Python in R Markdown

Part 1 Building our understanding

We will read one of the spectrum files to learn how to access dataframes in both sessions using `read_excel()` from `readxl`:

Insert R chunk **=> Run Chunk**

```
```{r}
##########   R    ##########
spectrum <- read_excel("Piano/01 Ballade No. 1, Op. 23_segments.xlsx")
```
```

Check the dimensions of the dataframe

How: Using Python in R Markdown

Part 1 Building our understanding

Modify R Chunk **=> Run Chunk**

````
```{r}
##########   R    ##########
spectrum <- read_excel("Piano/01 Ballade No. 1, Op. 23_segments.xlsx")
dim(spectrum)
```
````

```
[1] 100 5001
```

How: Using Python in R Markdown

Part 1 Building our understanding

Can you access the `spectrum` dataframe in a Python chunk and find out what type of object it is in the Python session?

Hint: the dataframe is `r.spectrum`

Insert Python chunk **=> Run Chunk**

```
```{python}
########## PYTHON ##########
type(r.spectrun)
```

<class 'pandas.core.frame.DataFrame'>
```

How: Using Python in R Markdown

Part 1 Building our understanding

It is a `Pandas`[1] dataframe.

Pandas is a module that provides dataframe data structures and analysis tools for working with them. A Pandas dataframe is just like an R dataframe.

In R, we access the column of a dataframe with `dataframe$colname`

In Python we access a Pandas dataframe column with `dataframe["colname"]`

[1] McKinney, W. (2010) Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56

How: Using Python in R Markdown

Part 1 Building our understanding

Can you use Python's `shape` in a new python chunk to determine the dimensions of the dataframe?

Hint: Python methods are aplied with the dot notation

Insert Python chunk **=> Run Chunk**

```
```{python}
########## PYTHON ##########
r.spectrum.shape
```

(100, 5001)
```

That's right!

How: Using Python in R Markdown

Part 1 Building our understanding

- 100 rows *i.e.*, segments.
- 5001 columns. The first column (curiously named "...1") is the segment label; the others are the 5000 features. They have the names "0" to "4999"

Using R to find the mean and s.d. of the second feature[1]

Insert R Chunk **=> Run Chunk**

```
```{r}
##########   R    ##########
mean(spectrum$`1`)
sd(spectrum$`1`)
```
```

```
[1] 4422.34
[1] 5711.785
```

[1] There's nothing special about feature 2. I'm avoiding the first feature simply because it is atypical

How: Using Python in R Markdown

Part 1 Building our understanding

Can you use Python to achieve the same:

Hint: the columns of pandas dataframes are accessed with `dataframe["colname"]`

Insert Python Chunk => Run Chunk

````
```{python}
########## PYTHON ##########
r.spectrum["1"].mean()
r.spectrum["1"].std()
```
````

```
4422.34
5711.784974979717
```

Well done!

How: Using Python in R Markdown

Part 1 Building our understanding

Knit your Rmd

Now is a good time to hit knit!

How: Using Python in R Markdown

Part 1 Building our understanding

Now read in the same file using Python. We will use Python in the second part of the tutorial to read in all our data.

You'll need to first add the `Pandas` module to your list of import statements in the Python chunk called mods

Modify Python Chunk **=> Run Chunk**

```
```{python mods}
########## PYTHON ##########
import os
import pandas as pd
```
```

How: Using Python in R Markdown

Part 1 Building our understanding

Read in the file:

Insert Python Chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
python_spectrum = pd.read_excel("Piano/01 Ballade No. 1, Op. 23_segments.xlsx")
```
````

How: Using Python in R Markdown

Part 2 Classification of audio data

Data import

This will be carried out with Python.

Overivew
The nine "_segments.xlsx" files are read in to a single Pandas dataframe, `df_seg`.

This requires nested `for` loops to iterate through the directories and through the files in the directories.

The information about the segments in "_SegmentInfo.xlsx" files are similarly read into a Pandas dataframe, `df_info` and a column is added to capture the instrument labeling.

You should be able to use and run the Python code without understanding how it works.

How: Using Python in R Markdown

Part 2 Classification of audio data

Data import

Copy the importing code from: Classifiying_music.html and paste the code in to a Python chunk

Insert Python Chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
dirs_to_use = ["Violin", "Piano", "Violin_and_Piano"]

df_seg = None
df_info = None
for d in dirs_to_use:
    for f in os.listdir(d):
    ... ... ...
    ... ... ... etc

```
````

How: Using Python in R Markdown

Part 2 Classification of audio data

At this point you might want to verify you have the pandas dataframe.

You can use `repl_python()` in the console

```
>  repl_python()
Python 3.7.3 (C:\PROGRA~3\ANACON~1\python.exe)
Reticulate 1.12.0.9003 REPL -- A Python interpreter in R
>>> type(df_seg)
<class 'pandas.core.frame.DataFrame'>
>>> df_seg.shape
(496, 5001)
>>> type(df_info)
<class 'pandas.core.frame.DataFrame'>
>>> df_info.shape
(496, 6)
```

How: Using Python in R Markdown

Part 2 Classification of audio data

Or add a chunk

Insert Python Chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
type(df_seg)
df_seg.shape
type(df_info)
df_info.shape
```
````

```
<class 'pandas.core.frame.DataFrame'>
(496, 5001)
<class 'pandas.core.frame.DataFrame'>
(496, 6)
```

How: Using Python in R Markdown

Part 2 Classification of audio data

Knit your Rmd

Now is good time to hit knit!

How: Using Python in R Markdown

Part 2 Classification of audio data

Principal Component Analysis (PCA) in Python

You'll need to first add PCA from the `scikit-learn`[1] module to your list of import statements in the Python chunk called mods

Modify Python Chunk **=> Run added line**

```
```{python mods}
########## PYTHON ##########
import os
import pandas as pd

# for PCA
from sklearn.decomposition.pca import PCA
```
```

[1] Pedregosa et al. (2011) Scikit-learn: Machine Learning in Python. JMLR 12: 2825-2830 http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

How: Using Python in R Markdown

Part 2: PCA in Python

The PCA is carried out with

Insert Python Chunk **=> Run Chunk**

````
```{python}
########## PYTHON ##########
# Apply PCA
mdl = PCA()
new_data = mdl.fit_transform(df_seg)

```
````

`new_data` contains the Principle Component scores

How: Using Python in R Markdown

Part 2: Visualising the PCA

First using Python. You'll need to add the `matplotlib.pyplot`[1] module to your list of import statements in the Python chunk called mods

Modify Python Chunk **=> Run added line**

```
```{python mods}
########## PYTHON ##########
import os
import pandas as pd

# for PCA
from sklearn.decomposition.pca import PCA
# for plotting
import matplotlib.pyplot as plt
```
```

[1] Hunter, J. D. (2007). "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3.

How: Using Python in R Markdown

Part 2: Visualising the PCA

First using Python

Copy the Biplot code from: Classifiying_music.html

Insert a new Python chunk and paste the code in

Insert Python Chunk **=> Run Chunk**

```
```{python}
########## PYTHON ##########
p = df_info["Instrument"] == "Piano"
v = df_info["Instrument"] == "Violin"
pv = df_info["Instrument"] == "Violin_and_Piano"
plt.figure()
... ... ...
... ... ... etc

```
```

You should see the Python plot in your file!

How: Using Python in R Markdown

Part 2: Visualising the PCA

Knit your Rmd

How: Using Python in R Markdown

Part 2: Visualising the PCA

Now using R!

Insert a new R chunk:

```
```{r}                                                    => Run Chunk
##########   R    ##########
df <- data.frame(pca1 = py$new_data[ ,1],
                 pca2 = py$new_data[ ,2],
                 instrument = py$df_info$Instrument)

ggplot(data = df, aes(x = pca1, y = pca2, color = instrument)) +
  geom_point()
```
```

You should see the R plot in your file!

How: Using Python in R Markdown

Part 2: Visualising the PCA

Knit your Rmd

Epic! Python and R plots in one doc!

The End, Congratulations!

By the end of the tutorial you should be able to:

- pass objects between simultaneous R and Python sessions
- use Python methods in R code
- incorporate Python snippets in to your R workflow even if you don't understand in detail how they work

Python scripts

- tutorial_code_sklearn.py
  Lots more machine learning methods applied to this data
- SoundSegment.py
  The class that processed audio
- generateSegments.py
  script that calls SoundSegment.py and outputs all the xlsx files

References and credits

- Allaire, J.J., Xie, Y., McPherson, J., Luraschi, J., Ushey, K., Atkins, A., Wickham, H., Cheng, J., Chang, W. and Iannone, R. (2019). rmarkdown: Dynamic Documents for R. R package version 1.12. URL https://rmarkdown.rstudio.com.
- Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, http://www.scipy.org/
- Kaggle.com. (2019). Kaggle Machine Learning & Data Science Survey 2017. [online] Available at: https://www.kaggle.com/kaggle/kaggle-survey-2017 [Accessed 5 Jul. 2019].
- Kaggle.com. (2019). Predicting R vs. Python | Kaggle. [online] Available at: https://www.kaggle.com/nanomathias/predicting-r-vs-python/notebook [Accessed 5 Jul. 2019].
- McKinney, W. (2010) Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56
- Mangano, A. (2019). The Reticulate Package Solves the Hardest Problem in Data Science: People. [Blog] R Views. Available at: https://rviews.rstudio.com/2019/03/18/the-reticulate-package-solves-the-hardest-problem-in-data-science-people/ [Accessed 5 Jul. 2019].
- Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. JMLR 12: 2825-2830 http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html
- Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org
- R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.
- Ushey, Kevin, J. J. Allaire, and Yuan Tang. (2019). "Reticulate: Interface to 'Python.'" https://CRAN.R-project.org/package=reticulate.
- Xie, Y. and Allaire J.J. and Grolemund, G. (2018). R Markdown: The Definitive Guide. Chapman and Hall/CRC. ISBN 9781138359338. URL https://bookdown.org/yihui/rmarkdown.
- Xie, Y. (2019). xaringan: Presentation Ninja. R package version 0.9. https://CRAN.R-project.org/package=xaringan