

[About](#) [Projects](#) [Talks](#) [Blog](#) [Links](#)

## Combining R and Python with {reticulate} and Quarto

Sometimes you might need to use R. Sometimes you might need to use Python. Sometimes you need to use both at the same time. This blog post shows you how to combine R and Python code using {reticulate} and output the results using Quarto.

January 6, 2023

The R versus Python debate has been going on for as long as both languages have existed. I'm not one to take sides - I think you need to use the best tool for the job. Sometimes R will be better. Sometimes Python will be better. But what happens if you need both languages in the same workflow? Do you need to choose? No, is the simple answer. You can use both. This blog post will show you how you can combine R and Python code in the same analysis using {reticulate} and output the results using Quarto.

## What is Quarto?

[Quarto](#) is an open-source scientific and technical publishing system that lets you combine narrative text with code to create reproducible and elegantly formatted output. If you're familiar with R Markdown, Quarto might sound somewhat similar - you can think of Quarto as next generation of R Markdown. The great thing about Quarto is that it doesn't just support code written in R - it supports other languages, including Python!

Quarto documents have two main sections: (i) the YAML header, where we specify document-wide properties such as the output format, and (ii) the content, which can include text, images, code, and more. For the example in this blog post, I'm going to output the code and results from the analysis to revealjs slides so my YAML header looks like this:

Copy

```
1 ---
2 title: "Modelling lemur weights with R and Python"
3 author: "Nicola Rennie"
4 format:
5   nr-revealjs:
6     embed-resources: true
7 ---
```

Your YAML might look a little different if you don't have a custom theme (another blog post on customising Quarto outputs will be on the way shortly!):

Copy

```
1 ---
2 title: "Modelling lemur weights with R and Python"
3 author: "Nicola Rennie"
4 format:
5   revealjs:
6     embed-resources: true
7 ---
```

If you want to output them to a PDF instead, then change the format argument in the YAML to be pdf:

Copy

```
1 ---
2 title: "Modelling lemur weights with R and Python"
3 author: "Nicola Rennie"
4 format: pdf
5 ---
```

## What is {reticulate}?

The [{reticulate}](#) [R package](#) provides an interface between R and Python which allows you to call Python code within R and translate between R and Python objects.



Using {reticulate} means that your Python code can access objects created in R, and vice versa. It means you can include Python code within an R package. It means you can run Python code in the RStudio IDE console.

## Using R and Python together

So let's see an example of how we can use R and Python within the same analysis workflow. We're going to read in a data set using R, fit a linear model in Python, then plot the residuals using {ggplot2} in R (and output our analysis as a set of slides).

First of all, we need some data. For this example, we're going to use the lemur data set. You can download the data from the [#TidyTuesday repository](#). This data set gives information on different lemurs living at [Duke Lemur Center](#), including their ages and weights measured over time. In this blog post, we're going to look at the relationship between the age and weight of lemurs.

Within our Quarto document, we start off with an R code block that reads in the data:

Copy

```
1 ```{r}
2 #| label: read-data
3 #| echo: true
4 #| message: false
5 #| cache: true
6 lemurs <- readr::read_csv('https://raw.githubusercontent.com/rfordatascience/tidyuesday/master/data/2021/2021-08-24/lemur_data.csv')
7 ```
```

Here, we've set four options for the code block: (i) labelled the code block with `label: read-data`, (ii) set `echo: true` to make my code show in the output, (iii) set `message: false` to make sure the messages from `read_csv()` don't show in the output, and (iv) set `cache: true` to cache the reading in of the data since the data set is reasonably large.

Now, we can include a second R code block to perform some data wrangling:

Copy

```
1 ```{r}
2 #| label: wrangling
3 #| echo: true
4 #| message: false
5 #| output-location: slide
6 library(dplyr)
7 library(knitr)
8 lemur_data <- lemurs %>%
9   filter(taxon == "ECOL",
10          sex == "M",
11          age_category == "adult") %>%
12   select(c(age_at_wt_mo, weight_g)) %>%
13   rename(Age = age_at_wt_mo,
14          Weight = weight_g)
15 kable(head(lemur_data))
16 ```
```

Here, we've selected only adult male collared brown lemurs, and chosen only the columns we want to model: age and weight. Setting `output-location: slide` puts the table we generate onto the following slide (since the code takes up most of the space on the slide).

So far, this has all been pretty standard. It's just some R code in a Quarto document. Now, we can add a Python code block to fit a model:

Copy

```
1 ```{python}
2 #| label: modelling
3 #| echo: true
4 #| message: false
5 lemur_data_py = r.lemur_data
```

```

6
7 import statsmodels.api as sm
8 y = lemur_data_py[["Weight"]]
9 x = lemur_data_py[["Age"]]
10 x = sm.add_constant(x)
11 mod = sm.OLS(y, x).fit()
12 lemur_data_py["Predicted"] = mod.predict(x)
13 lemur_data_py["Residuals"] = mod.resid
14 ```

```

The key part here is the `r.` on line 4 - this tells {reticulate} to look in the R code for an object called `lemur_data`. You can think of this line as importing your data from R to Python. Other than that, it's pretty standard Python code. Here, we've kept it very simple and fitted a linear model:

$$weight_i = \beta_0 + \beta_1 age_i + e_i$$

Obviously, R has the capabilities to fit a linear model, and for this simple example there's no *need* for us to use Python. However, this illustrates how you might approach the problem, if you find yourself needing to use a particular library or model that you can only find in Python.

After we've fitted any type of model, we need to check the output. For linear regression, one of the most common methods of model inspection is to plot and analyse the residuals. Residuals tell us how far away a point is from the regression line. For linear models, the main assumption is that the errors are independent and normally distributed. If our data fits that assumption, we should see that:

- the residuals are symmetric above and below 0;
- we don't see any patterns in the residuals as the predicted values increase;
- most of the points are close to 0, and there are fewer points with higher magnitude of residual.

Whilst you can plot residuals in Python quite easily, in my opinion it's hard to beat the quality of graphics you can achieve with {ggplot2} in R. So we include a final R code block to plot the residuals:

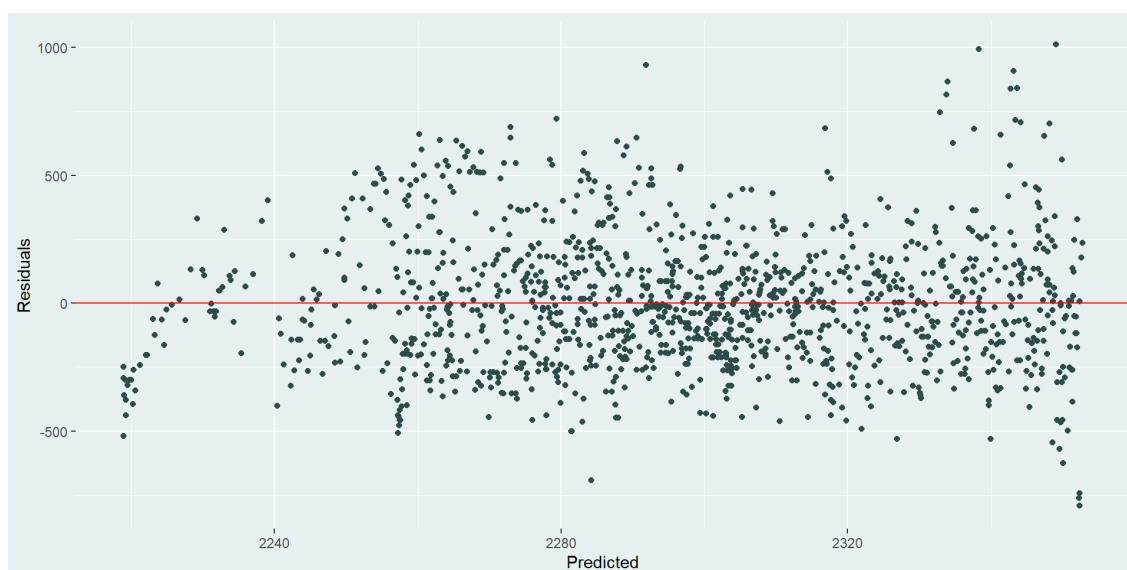
Copy

```

1 ```{r}
2 #| label: plotting
3 #| echo: true
4 #| output-location: slide
5 #| message: false
6 #| fig-align: center
7 #| fig-alt: "Scatter plot of predicted and residual values for the fitted linear model."
8 library(reticulate)
9 library(ggplot2)
10 lemur_residuals <- py$lemur_data_py
11 ggplot(data = lemur_residuals,
12       mapping = aes(x = Predicted,
13                     y = Residuals)) +
14   geom_point(colour = "#2F4F4F") +
15   geom_hline(yintercept = 0,
16             colour = "red") +
17   theme(panel.background = element_rect(fill = "#eaf2f2",
18                                         colour = "#eaf2f2"),
19         plot.background = element_rect(fill = "#eaf2f2",
20                                         colour = "#eaf2f2"))
21 ```

```

Here, we used `py$` to import our data and specify that we want to use `lemur_data_py` from our Python code block. Our R code block returns the following residual plot:



Our residuals look okay - they're not great. It looks like there's still some unaccounted-for trend for lemurs with smaller predicted weights. Maybe they're not growing at a linear rate. It also looks like there is an increase in variation for lemurs with large predicted weights. It would be useful to go back and try a few other models here to see what happens, but we'll leave that for another day...

## How does it work?

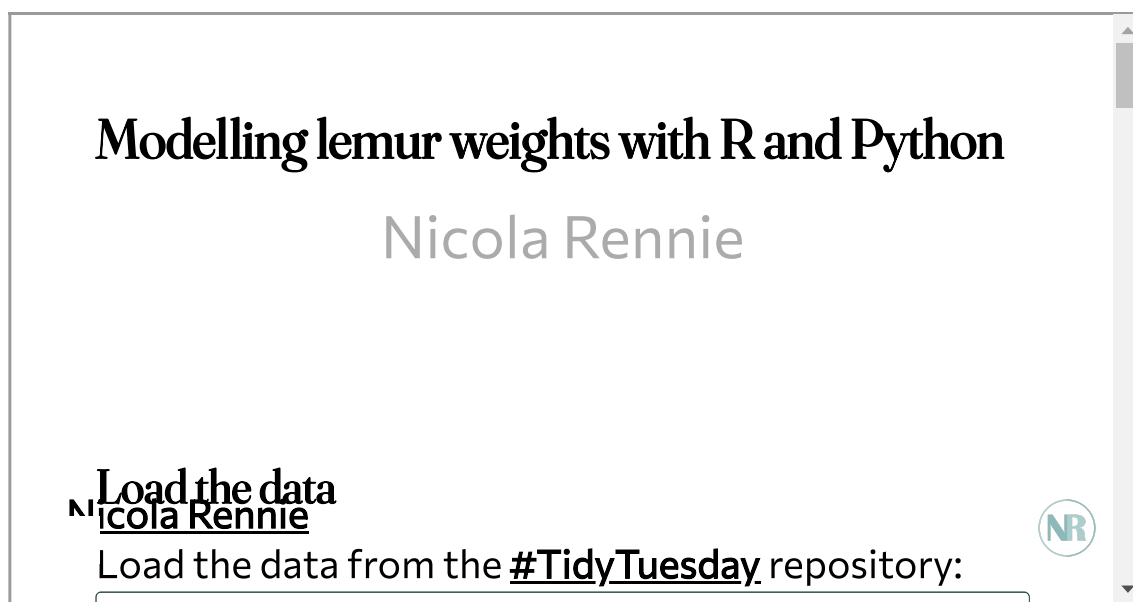
There are a couple of elements to the code that means R and Python *just works* together. Since the document starts with an R code block, the engine used to render the document will be knitr. The rest of this blog post assumes we're using a knitr engine, so if you've specified a Jupyter engine instead, this method won't work. The [Quarto documentation](#) gives a bit more detail on how the engine is chosen. Note that these examples also work with R Markdown (not just Quarto) - although the YAML and code block options will need to be changed.

Normally, if you're rendering a Quarto document that contains only Python code blocks, it will render using a Jupyter engine. That's not what's happening here. Here, the code block is still being rendered using knitr, but it knows to use {reticulate} to run the code when it finds a Python code block. In the Python code block, we accessed the data from the R code block using `r.lemur_data`. The `r.` is the key element here - it tells {reticulate} to look in the R code for an object called `lemur_data` and use that.

In the final R code block, we do something sort of similar to `py$lemur_data_py`. Here, we use `py$` to specify that we want to use `lemur_data_py` from the Python code block. You'll notice that we also need to explicitly call `library(reticulate)` for this to work (or use `reticulate::py`) - since it's an R code block, {knitr} doesn't know that it needs to use {reticulate} without us telling it to.

## The final output

Let's see what our slides containing the code, table, and plot look like...



► Show Quarto .qmd file

You can download the .qmd file from my [website](#).

## Additional resources

You can use {reticulate} outside of Quarto (and R Markdown) documents, including to run Python code from the console in RStudio. If you want to read more about what {reticulate} can do, the [package website](#) has a lot of nice examples and guidance.

If you're interested in using a Jupyter engine (likely if most of your code is in Python and you want to use a little bit of R), then you can do something similar using the [rpy2 module](#).

Now you're ready to use R and Python together!



Image: [giphy.com](#)

For attribution, please cite this work as:

**Combining R and Python with {reticulate} and Quarto.**  
Nicola Rennie. January 6, 2023.  
[nrennie.rbind.io/blog/combining-r-and-python-with-reticulate-and-quarto](https://nrennie.rbind.io/blog/combining-r-and-python-with-reticulate-and-quarto)

Licence: [creativecommons.org/licenses/by/4.0](https://creativecommons.org/licenses/by/4.0)

5 Comments - powered by utteranc.es

fjodor commented on 25. Jan. 2023

Thanks for the post! Nice to see some quarto tricks in action, e. g. output-location: slide.

👍 1

bassamsdata commented on 27. Apr. 2023

So, everytime I ask AI search chats about quarto and Python, it directs me to this blog article, which is fantastic. This is the third time I've looked for quarto and python and come upon this site. It's a fantastic blog article that the official quarto website should link to.  
Thank you for writing this great article.

👍 1

nrennie commented on 4. Mai 2023

Owner

Thanks - I'm glad you found it useful!

ndjrt commented on 15. Juni 2023

So when I try to run the qmd file, I get the following error for the r chunk where the reticulate library is loaded.

### Any suggestions to get it to work?

Error in `fortify()` :  
! data must be a <data.frame>, or an object coercible by `fortify()` , not a list.  
Backtrace:

- 1. `ggplot2::ggplot(...)`
- 2. `ggplot2:::ggplot.default(...)`
- 3. `ggplot2:::fortify.default(data, ...)`

nrennie commented on 19. Juli 2023

Owner

Hi @ndjrt, do you mean the code chunk labelled `label: plotting` ? I can't reproduce the error unfortunately. Is reticulate properly installed, and can you inspect the output from the python block to make sure it looks as you would expect?

Write

Preview

Sign in to comment

 Styling with Markdown is supported

Sign in with GitHub

