Python con R reticulate : : **GUÍA RÁPIDA**

El paquete **reticulate** permite usar Python y R juntos en código R, en documentos de R Markdown y en RStudio IDE.

Python en R Markdown

(Opcional) Construir un entorno (env) de Python para usar.

Agregar **knitr::knit_engines$set(python = reticulate::eng_python)** para configurar el trozo de código (*chunk*) y **reticulate** en Python (no es necesario para knitr >= 1.18).

Sugiere un entorno de Python para usar en la configuración del trozo de código.

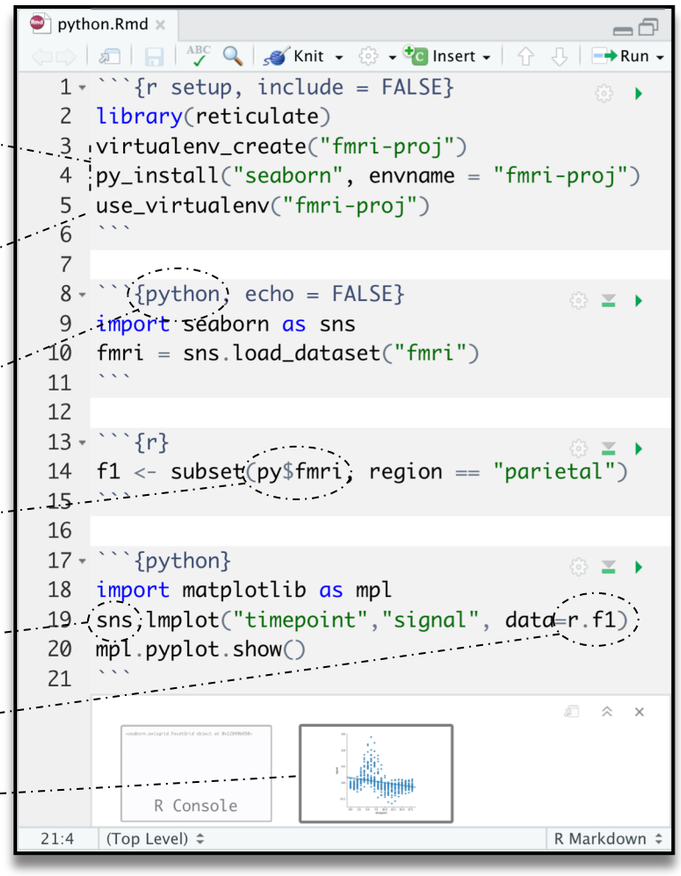
Comienza los trozos de código de Python con **```{python}**. Las opciones del trozo de código como **echo**, **include**, etc. ,funcionan como es esperado.

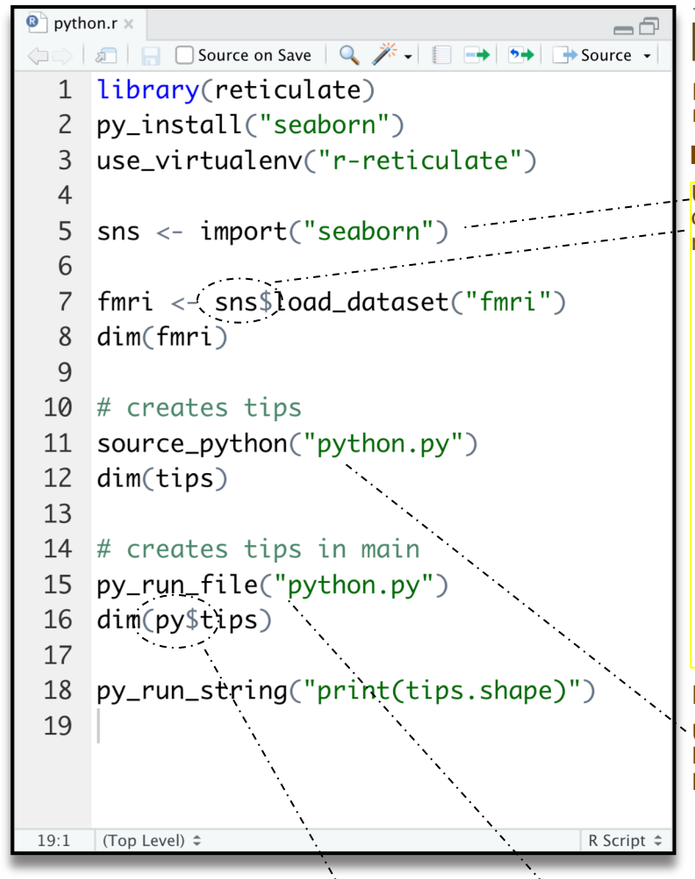
Usa el objeto **py** para acceder a los objetos creados en trozos de código de Python desde trozos de código de R.

Todos los trozos de código de Python se ejecutan con una única sesión de Python para que se pueda acceder a todos los objetos creados en trozos de código previos.

Usa el objeto **r** para acceder a los objetos creados en trozos de código en R desde los trozos de código en Python.

Las salidas se muestran debajo del trozo de código, Incluyendo los gráficos en la librería matplotlib.





Python en R

Llamar a Python desde el código R de tres maneras:

Importar módulos de PYTHON

Usa **import**() para importar cualquier módulo de Python. Accede a los atributos de un modulo con **$**.

- **import**(module, as = NULL, convert = TRUE, delay_load = FALSE) Import a Python module. Si convert = TRUE, Los objetos de Python se convierten a su equivalente en R. También **import_from_path**. *import("pandas")*
- **import_main**(convert = TRUE) Importa el módulo principal donde Python ejecuta su código por defecto. *import_main()*
- **import_builtins**(convert = TRUE) Importa las funciones integradas de Python *import_builtins()*

Fuente de archivos de PYTHON

Usa **source_python**() para obtener un script de Python y hacer que las funciones y objetos de Python queden disponibles en el entorno de R.

- **source_python**(file, envir = parent.frame(), convert = TRUE) Ejecuta un script de Python asignando objetos a un entorno específico de R. *source_python("file.py")*

Ejecutar código de PYTHON

Ejecutar código de Python en el módulo principal de Python con **py_run_file**() o **py_run_string**().

- **py_run_string**(code, local = FALSE, convert = TRUE) Ejecuta código de Python (pasado a una cadena de caractéres, *"string"*) en el módulo principal. *py_run_string("x = 10"); py$x*
- **py_run_file**(file, local = FALSE, convert = TRUE). Ejecuta archivos de Python en el módulo principal. *py_run_file("script.py")*
- **py_eval**(code, convert = TRUE) Ejecuta una expresión de Phyton y devuelve el resultado. También **py_call**. *py_eval("1 + 1")*

Acceder a los resultados y a cualquier otra cosa en el módulo principal de Python con **py**.

- **py** Un objeto R que contiene el módulo principal de Python y los resultados almacenados allí. *py$x*

Conversión de objetos

Consejo: para indexer objetos de Python que comienzan con 0, usar números enteros, por ej. 0L

Reticulate provee la conversion automática entre Python y R para muchos tipos de objetos de Python.

R ⟷	Python
Single-element vector	Scalar
Multi-element vector	List
List of multiple types	Tuple
Named list	Dict
Matrix/Array	NumPy ndarray
Data Frame	Pandas DataFrame
Function	Python function
NULL, TRUE, FALSE	None, True, False

O si se prefiere, se pueden convertir manualmente con:

py_to_r(x) Convierte un objeto de Python en un objeto de R. También **r_a_py**. *py_to_r(x)*

tuple(..., convert = FALSE) Crea una tupla (conjunto ordenado e inmutable de elementos del mismo o diferente tipo) en Python. *tuple("a", "b", "c")*

dict(..., convert = FALSE) Crea un diccionario de objetos de Python. También se usa **py_dict** para hacer un diccionario que usa objetos de Python como clave. *dict(foo = "bar", index = 42L)*

np_array(data, dtype = NULL, order = "C") Crea NumPy arrays. *np_array(c(1:8), dtype = "float16")*

array_reshape(x, dim, order = c("C", "F")) Remodela una matriz de Python. *x <- 1:4; array_reshape(x, c(2, 2))*

py_func(object) Esta función (*Wrap*) ajusta función de R en una función de Python con la misma firma. *py_func(xor)*

py_main_thread_func(object) Crea una función que siempre será llamada en el hilo principal.

iterate(..., convert = FALSE) Aplica una función de R a cada valor de iterador de Python o devuelve los valores como un vector de R, drenando el iterardor a medida que Avanza. También **iter_next y as_iterator**. *iterate(iter, print)*

py_iterator(fn, completed = NULL) Crea un iterador de Python desde una función de R .*seq_gen <- function(x){n <- x; function() {n <<- n + 1; n}};py_iterator(seq_gen(9))*

Ayudantes ("helpers")

py_capture_output(expr, type = c("stdout", "stderr")) Captura y devuelve la salida de Python. También **py_suppress_warnings**. *py_capture_output("x")*

py_get_attr(x, name, silent = FALSE) Toma un atributo de un objeto de Python. También **py_set_attr**, **py_has_attr**, y **py_list_attributes**. *py_get_attr(x)*

py_help(object) abre la página de documentación de un objeto de Python. *py_help(sns)*

py_last_error() Toma el último error encontrado en Python. También **py_clear_last_error** para limpiar el ultimo error. *py_last_error()*

py_save_object(object, filename, pickle = "pickle") Guarda y carga un objeto de Python con pickle. También **py_load_object**. *py_save_object(x, "x.pickle")*

with(data, expr, as = NULL, ...) Evalúa una expresión dentro de un contexto de administración de Python. *py <- import_builtins(); with(py$open("output.txt", "w") %as% file, { file$write("Hello, there!")})*

Python en IDE

Requiere reticulate y RStudio v1.2 o mayor.

Resaltado de sintaxis para scripts y trozos de código de Python

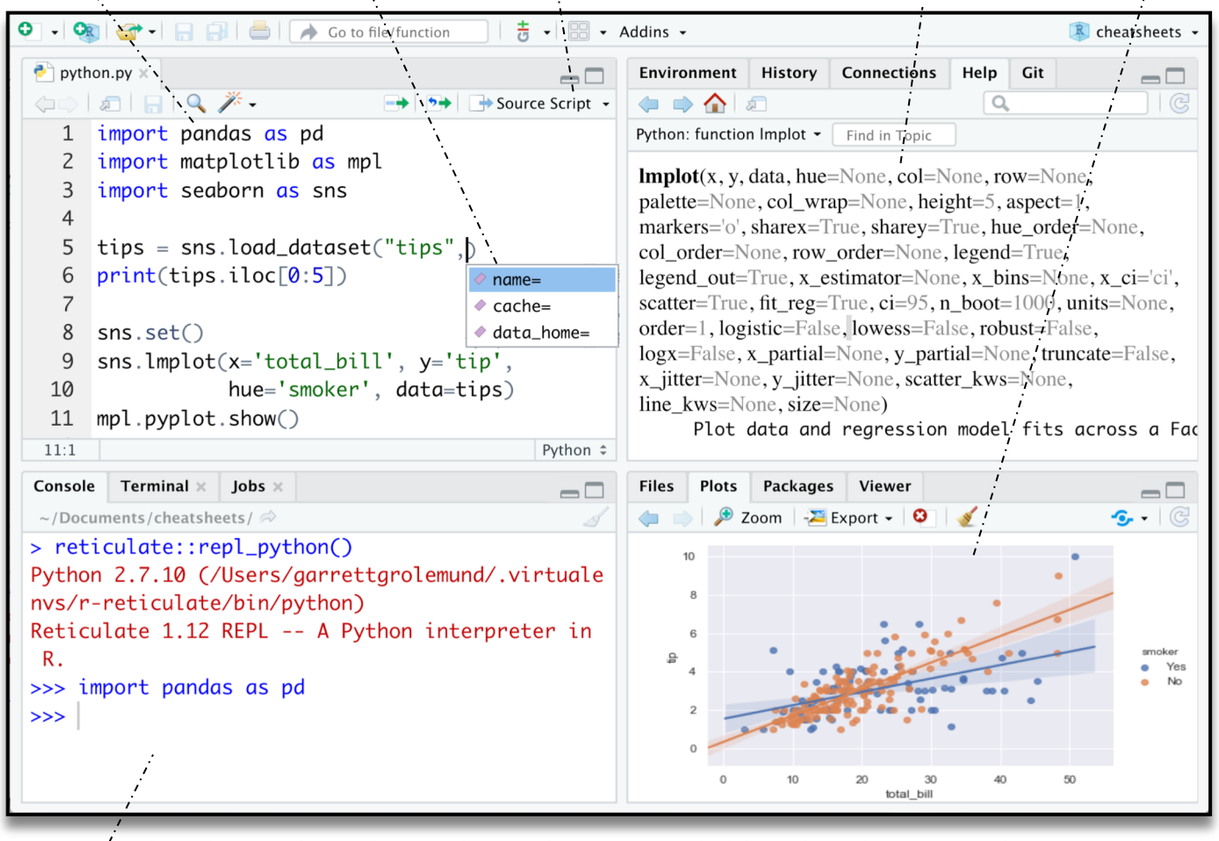
Tabulaciones para completar funciones y objetos de Python (y módulos de Python importados en scripts de R)

Fuente de scripts de Python

Ejecuta línea a línea un código de Python con **Cmd + Enter** (**Ctrl + Enter**)

Presione **F1** sobre un símbolo de Python para mostrar la ayuda para ese símbolo..

Los gráficos de *matplotlib* se muestran en el panel de gráficos.

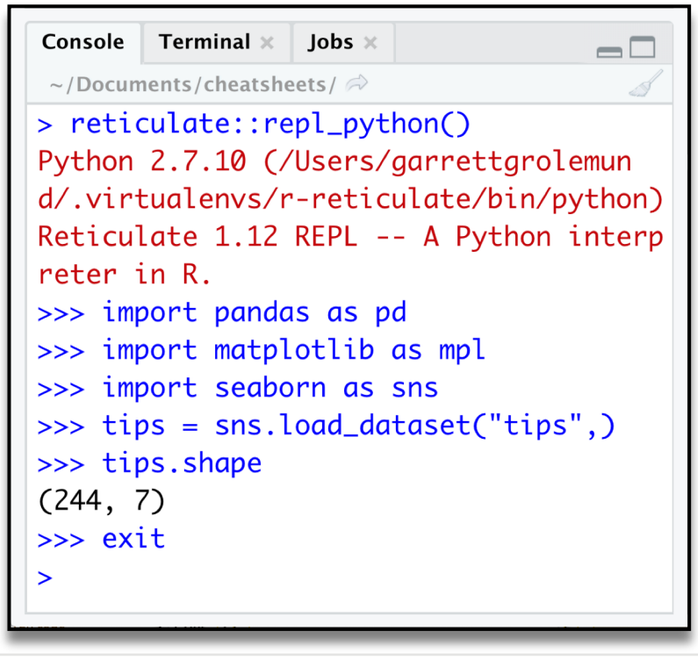


Un REPL de Python se abre en la consola cuando se ejecuta el código de Python con combinaciones de teclas. Tipear **exit** para cerrar.

Python REPL

El REPL (Read, Eval, Print Loop) es una línea de comando donde se puede ejecutar el código de Python y ver los resultados.

1. Abrir en la consola con **repl_python**(), o ejecutar el siguiente código en el script de Python: **Cmd + Enter** (**Ctrl + Enter**).
   - **repl_python**(module = NULL, quiet = getOption("reticulate.repl.quiet", default = FALSE)) Lanza una REPL de Python. Ejecutar **exit** para cerrar. *repl_python()*
2. Escriba commandos en el indicador **>>>** prompt
3. Presionar **Enter** para ejecutar el código
4. Tipear **exit** para cerrar y regresar a la consola de R



Configurar Python

Reticulate se une a una instancia local de Python cuando se llama por primera vez **import**() directa o implícitamente desde una sesión de R. Para controlar el proceso, buscar o crear la instancia de Python que se desee. Luego sugerir la instancia a reticulate. **Reiniciar R para desvincular.**

Encontrar Python

- **py_discover_config**() Devuelve todas las versiones de Python detectadas. Use **py_config** para chequear que versión ha sido cargada. *py_config()*
- **py_available**(initialize = FALSE) Chequear si Python esta disponible en su sistema. También **py_module_available**, **py_numpy_module.** *py_available()*
- **virtualenv_list**() Devuelve un listado de todos los *entornos virtuales* disponibles. También **virtualenv_root**(). *virtualenv_list()*
- **conda_list**(conda = "auto") Devuelve un listado de todos los entornos conda disponibles. También **conda_binary**() y **conda_version**(). *conda_list()*

Crear un entorno Python

- **virtualenv_create**(envname) Crea un nuevo entorno virtual. *virtualenv_create("r-pandas")*
- **conda_create**(envname, packages = NULL, conda = "auto") Crea un nuevo entorno Conda . *conda_create("r-pandas", packages = "pandas")*

Instalar paquetes

Instale paquetes de Python con R (debajo) o el shell:
pip install SciPy
conda install SciPy

- **py_install**(packages, envname = "r-reticulate", method = c("auto", "virtualenv", "conda"), conda = "auto", ...) Instala paquetes de Python en un entorno de Python nombrado "r-reticulate". *py_install("pandas")*
- **virtualenv_install**(envname, packages, ignore_installed = FALSE) Instala un paquete dentro de un entorno virtual. *virtualenv_install("r-pandas", packages = "pandas")*
- **virtualenv_remove**(envname, packages = NULL, confirm = interactive()) Remueve paquetes individuales o un entorno virtual entero .*virtualenv_remove("r-pandas", packages = "pandas")*
- **conda_install**(envname, packages, forge = TRUE, pip = FALSE, pip_ignore_installed = TRUE, conda = "auto") Instala un paquete dentro de un entorno Conda. *conda_install("r-pandas", packages = "plotly")*
- **conda_remove**(envname, packages = NULL, conda = "auto") Remueve paquetes individuales o un entorno Conda entero. *conda_remove("r-pandas", packages = "plotly")*

Sugerir un entorno

Para elegir una instancia de Python para enlazar, reticulate escanea las instancias en su computadora en orden, **deteniéndose en la primer instancia que contenga al módulo llamado por import()**.

1. Instancia a la que hace referencia al entorno variable **RETICULATE_PYTHON** (si es especificado). Consejo: establecer en el archivo .Renviron
   - **Sys.setenv**(RETICULATE_PYTHON = PATH) Establece por defecto Python binario. Persiste a través de las sesiones. Deshacer con **Sys.unsetenv**. *Sys.setenv(RETICULATE_PYTHON="/usr/local/bin/python")*
2. La instancia a la que hace referencia **use_functions** aplica si es llamada antes de **import()**. Esto fallará silenciosamente si es llamada antes de importer a menos que sea **required = TRUE**.
   - **use_python**(python, required = FALSE) Sugiere Python binario para usar *path*. *use_python("/usr/local/bin/python")*
   - **use_virtualenv**(virtualenv = NULL, required = FALSE) Sugiere un entorno virtual de Python *use_virtualenv("~/myenv")*
   - **use_condaenv**(condaenv = NULL, conda = "auto", required = FALSE) Sugiere usar un entorno Conda. *use_condaenv(condaenv = "r-nlp", conda = "/opt/anaconda3/bin/conda")*
3. Dentro de los entornos virtuales y conda que llevan el mismo nombre del módulo importado, p. ej.: *~/anaconda/envs/nltk* for *import("nltk")*
4. En la ubicación de Python binario descubierto en el Sistema PATH (i.e. **Sys.which("python")**).
5. En ubicaciones habituales para Python, p. Ej.: **/usr/ local/bin/python**, **/opt/local/bin/python**...