

# R Interface to Python



kilometer00

Who ! ?



Who ! ?

名前：三村 @kilometer

職業：ポストドク (こうがくはくし)

専門：行動神経科学 (靈長類)

脳イメージング

医療システム工学

R歴：～10年ぐらい

流行：カメ



- 2018.07.15 Tokyo.R #71  
Landscape with R – the Japanese R community
- 2018.10.20 Tokyo.R #73  
BeginneR Session – Visualization & Plot
- 2019.01.19 Tokyo.R #75  
BeginneR Session – Data pipeline
- 2019.03.02 Tokyo.R #76  
BeginneR Session – Data pipeline
- 2019.04.13 Tokyo.R #77  
BeginneR Session – Data analysis
- 2019.05.25 Tokyo.R #78  
BeginneR Session – Data analysis
- 2019.06.29 Tokyo.R #79  
BeginneR Session – 確率の基礎

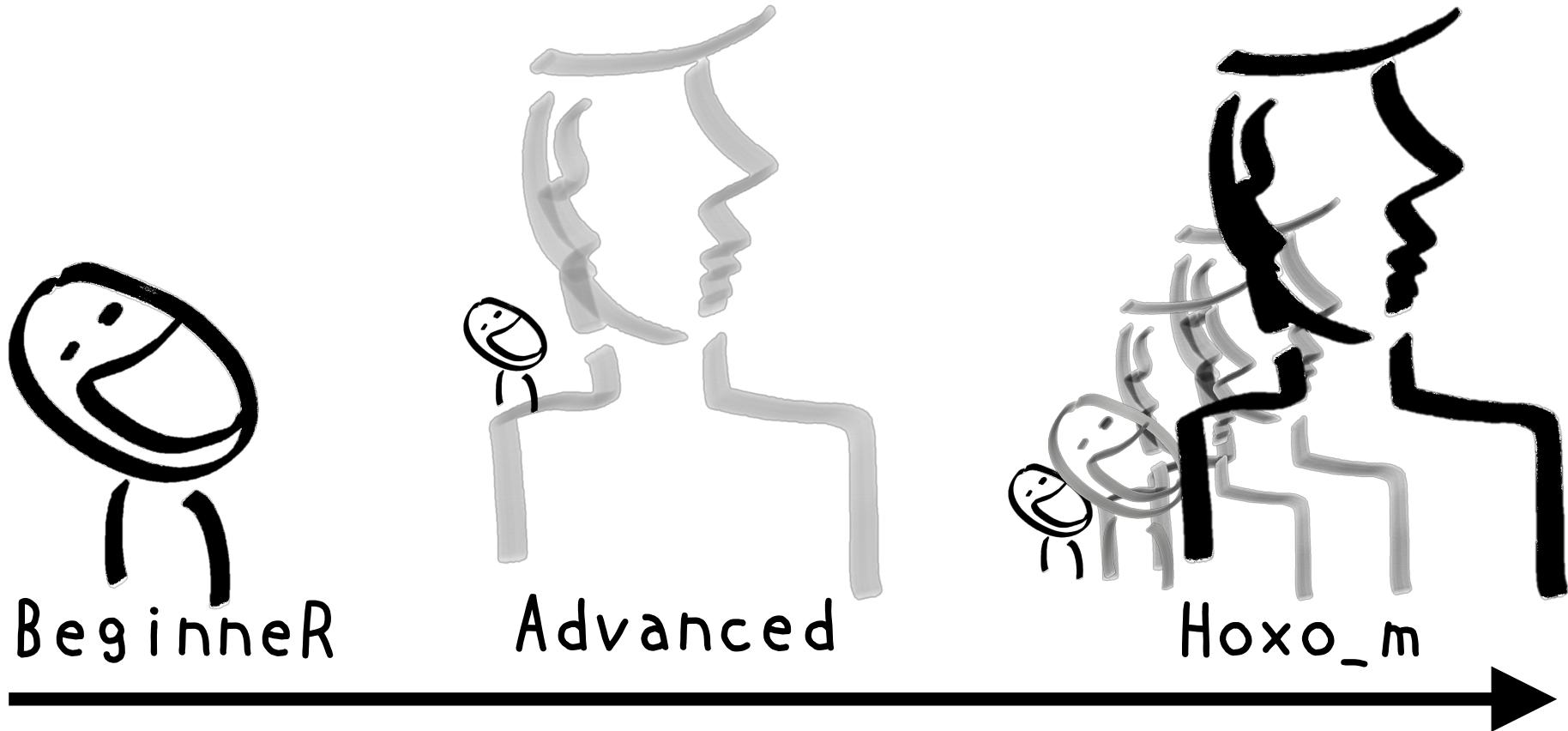
# BeginneR Session



Before



After



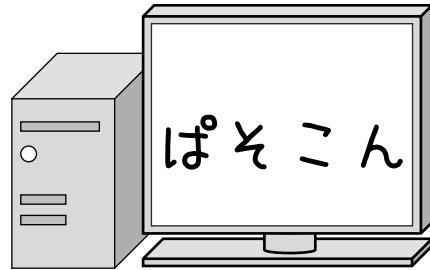
If I have seen further it is by standing on  
the shoulders of Giants.

-- Sir Isaac Newton, 1676

# R Interface to Python

# —昔前

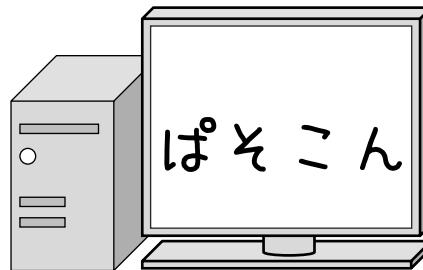
私もそろそろ



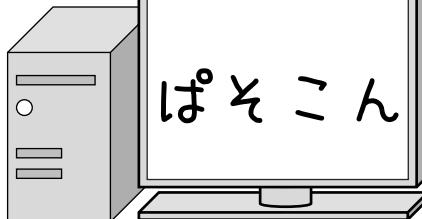
を、始めたいんだ"けど。

できるんで"しょ？教えてよ。

いいけど、



で何がしたいの？

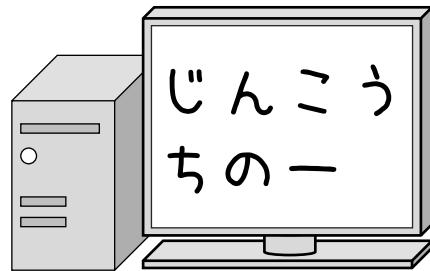


ぱそこんって何でもできるんで"しょ？

でも、どうしたらいいか分かんないんだ。

最近

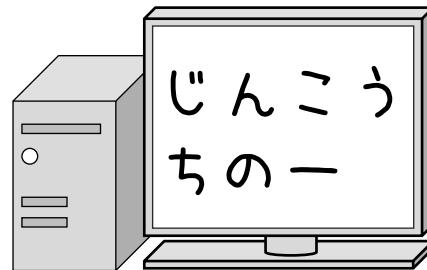
私もそろそろ



を、始めたいんだけど。

できるんでしょ？教えてよ。

いいけど、



で何がしたいの？

じんこう  
ちの一

って何でもできるんでしょ？

でも、どうしたらいいか分かんないんだ。

私もそろそろ



を、始めたいんだけど。

できるんでしょ？教えてよ。

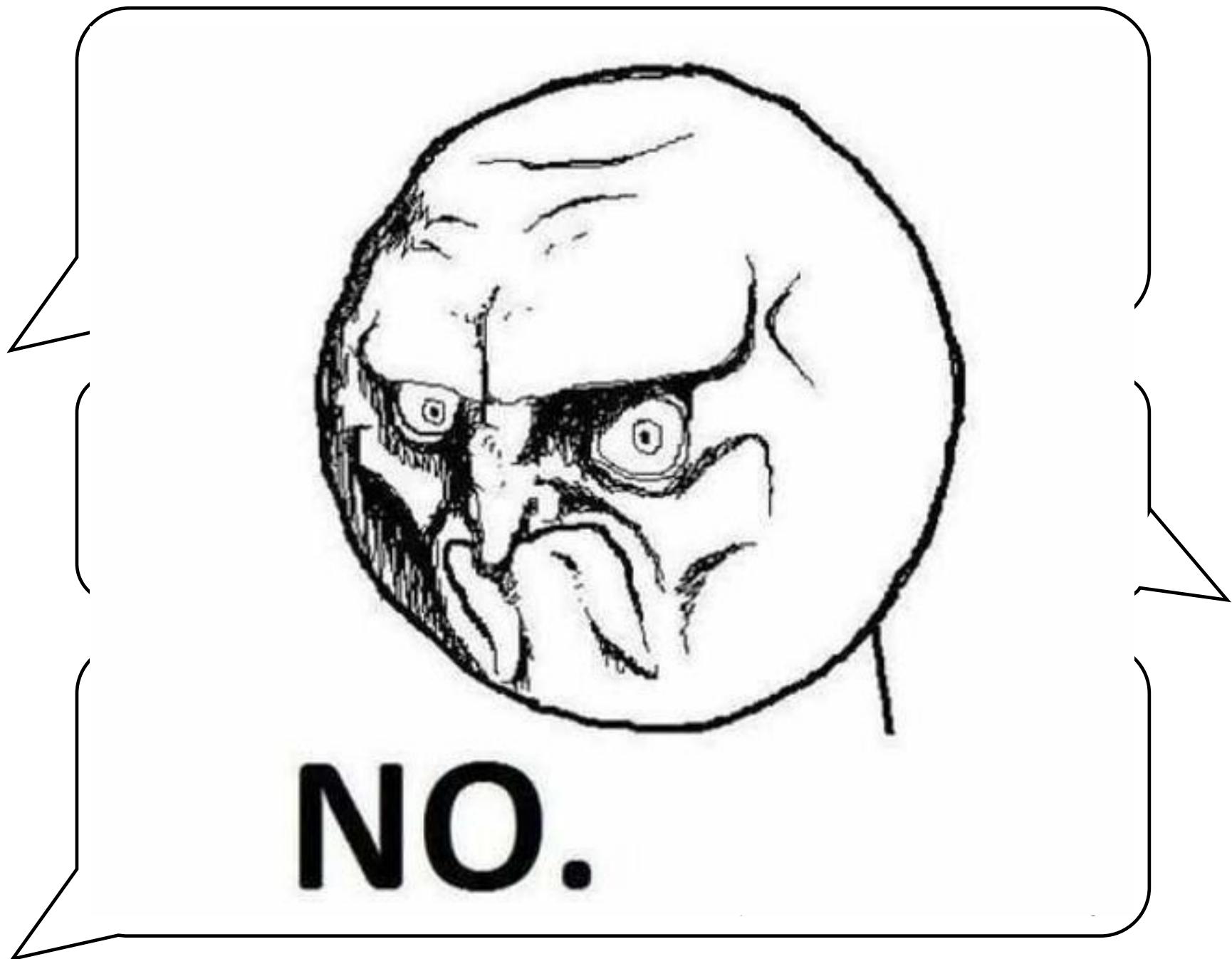
いいけど、



で何がしたいの？

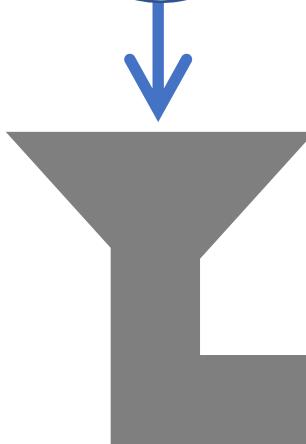
って何でもできるんでしょ？

でも、どうしたらいいか分かんないんだ。

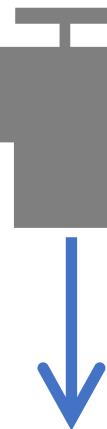
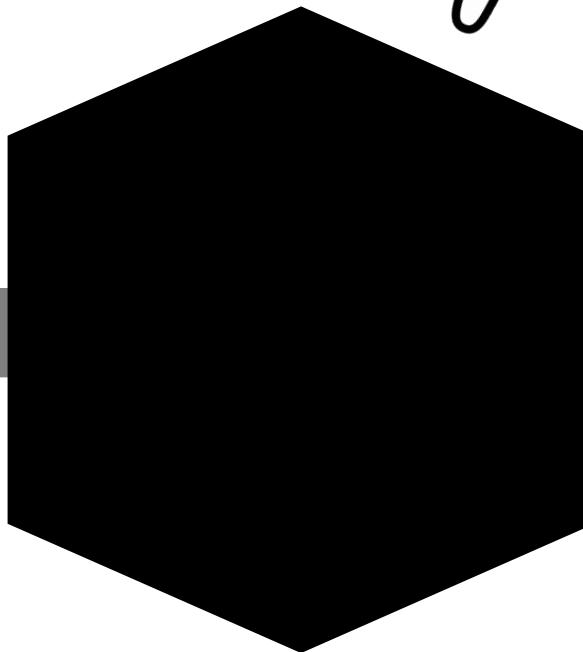


**NO.**

**Input**

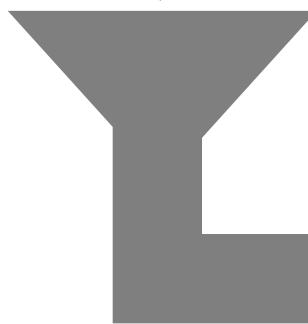


**Do NOT start from here**

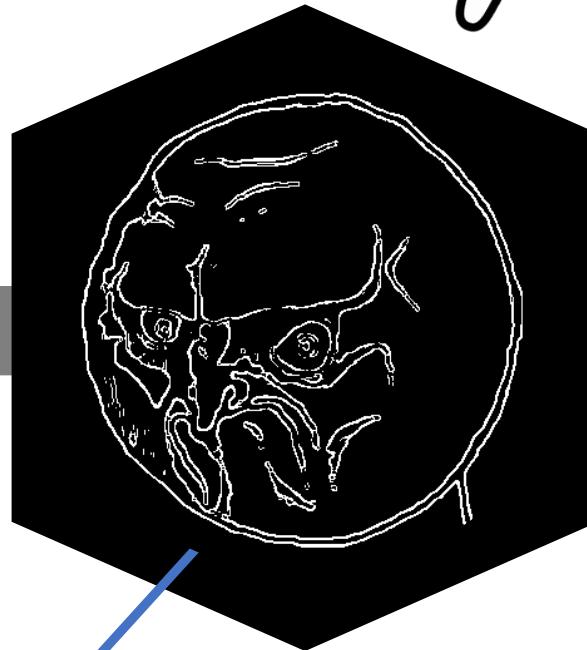


**Output**

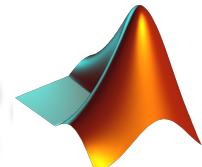
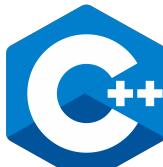
Input



Do NOT start from here



Whatever



...

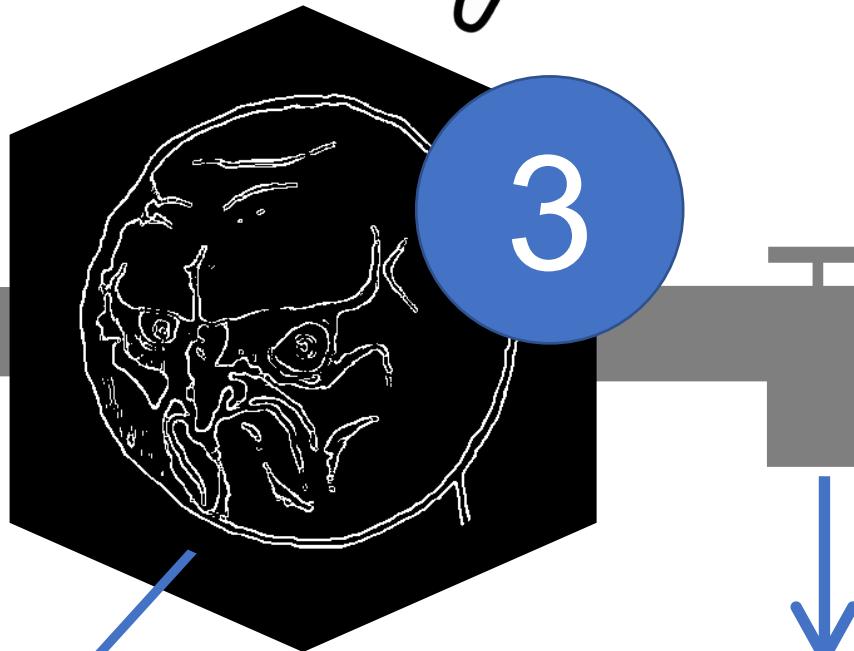
Output

**Input**

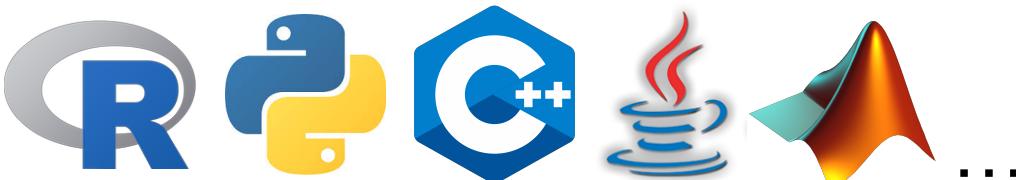
2



Do NOT start from here



Whatever

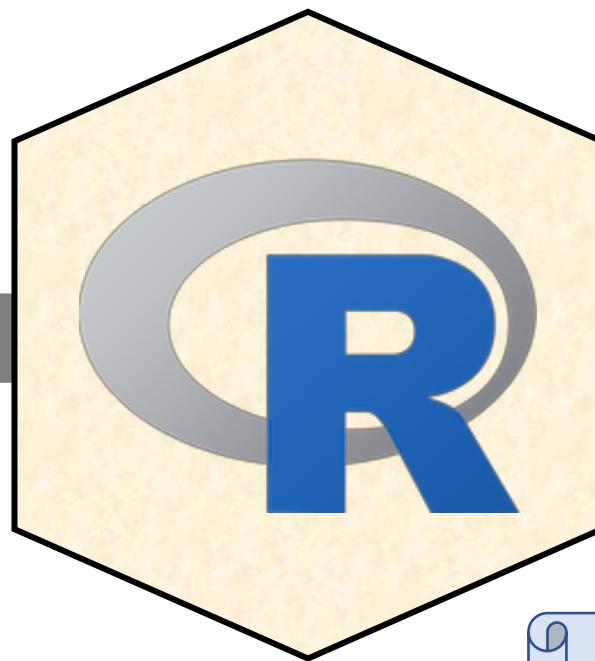


1



Output

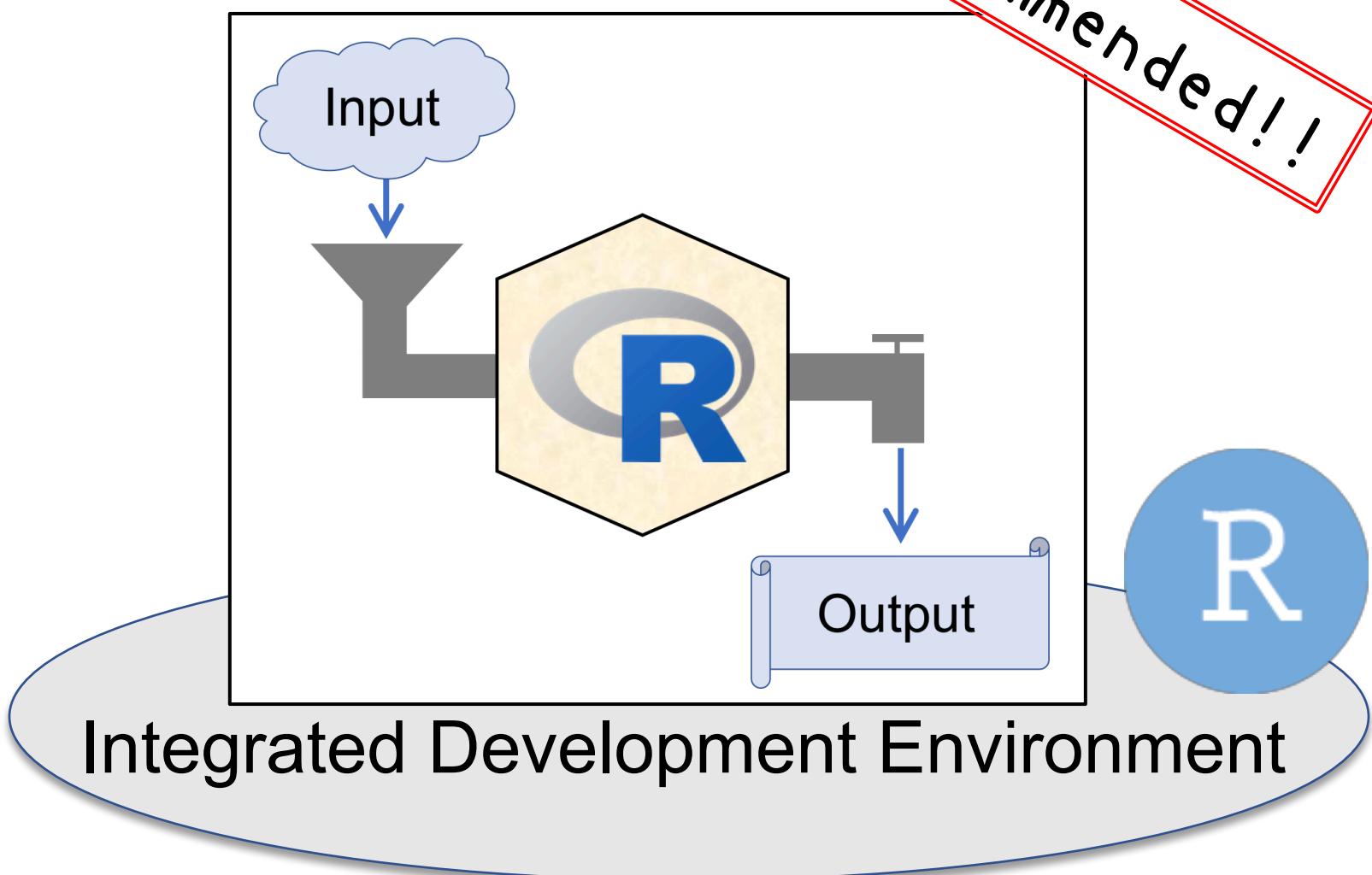
**Input**



**Output**

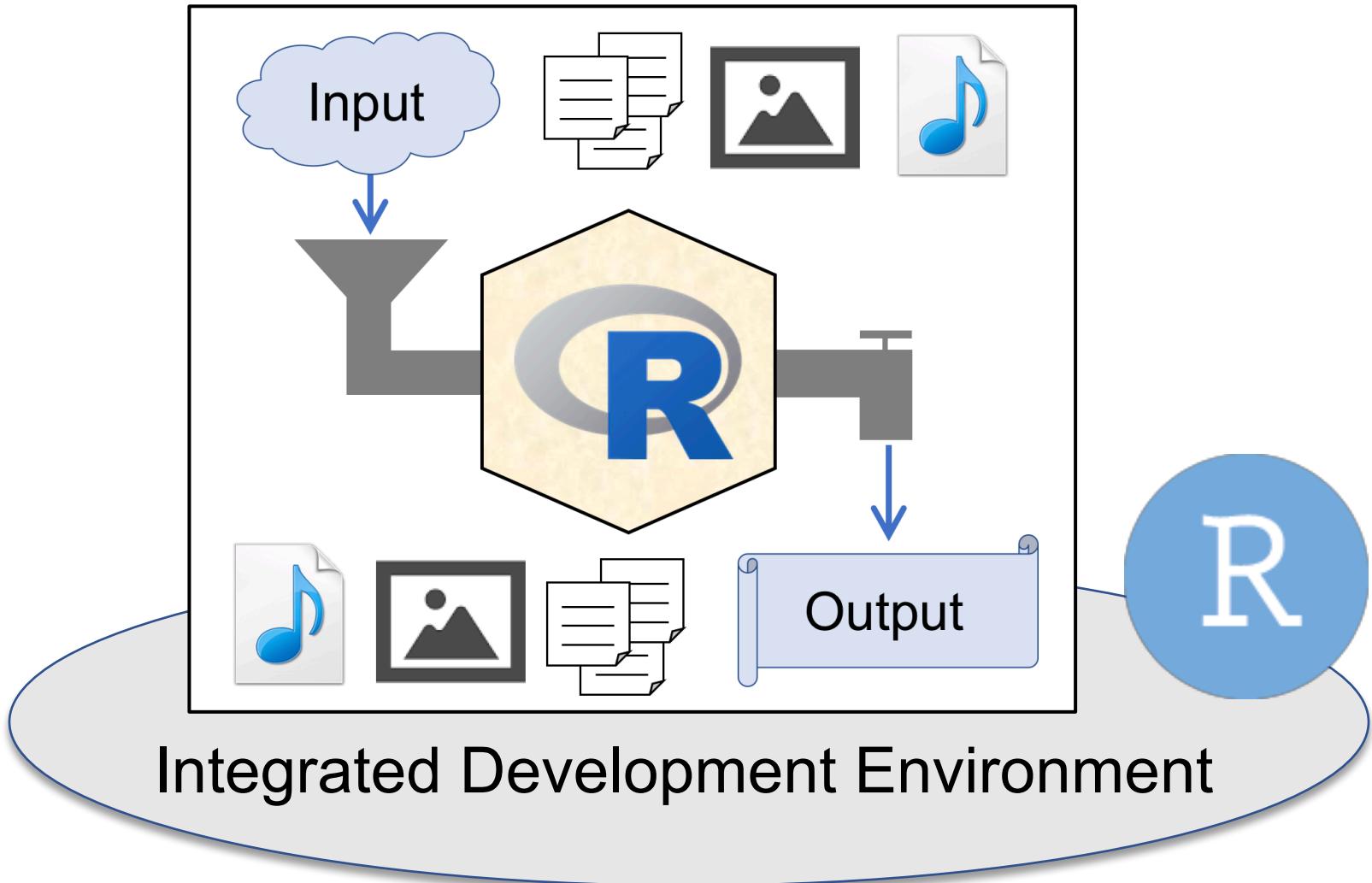
# RStudio

<https://www.rstudio.com/>

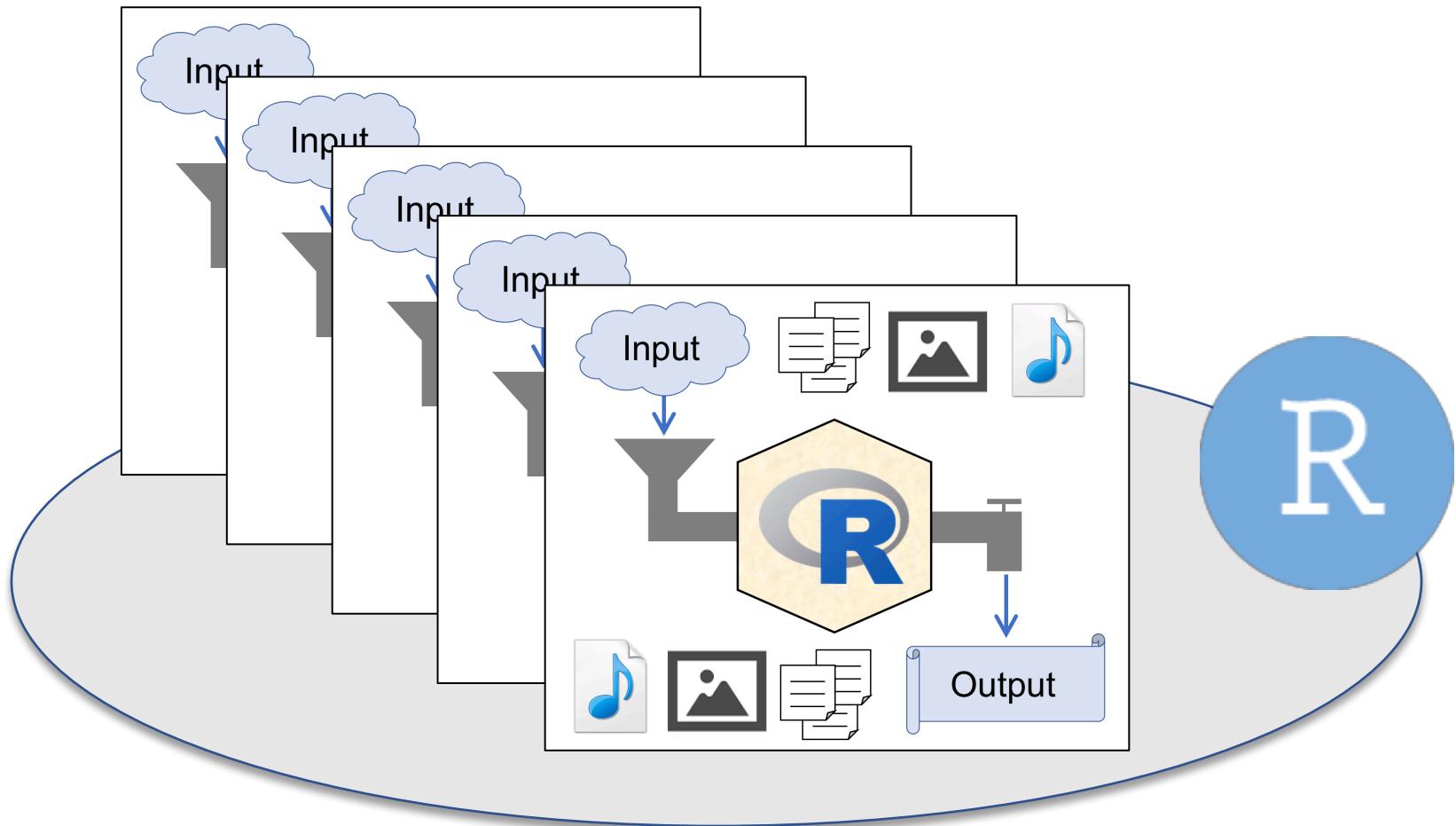


# RStudio

<https://www.rstudio.com/>

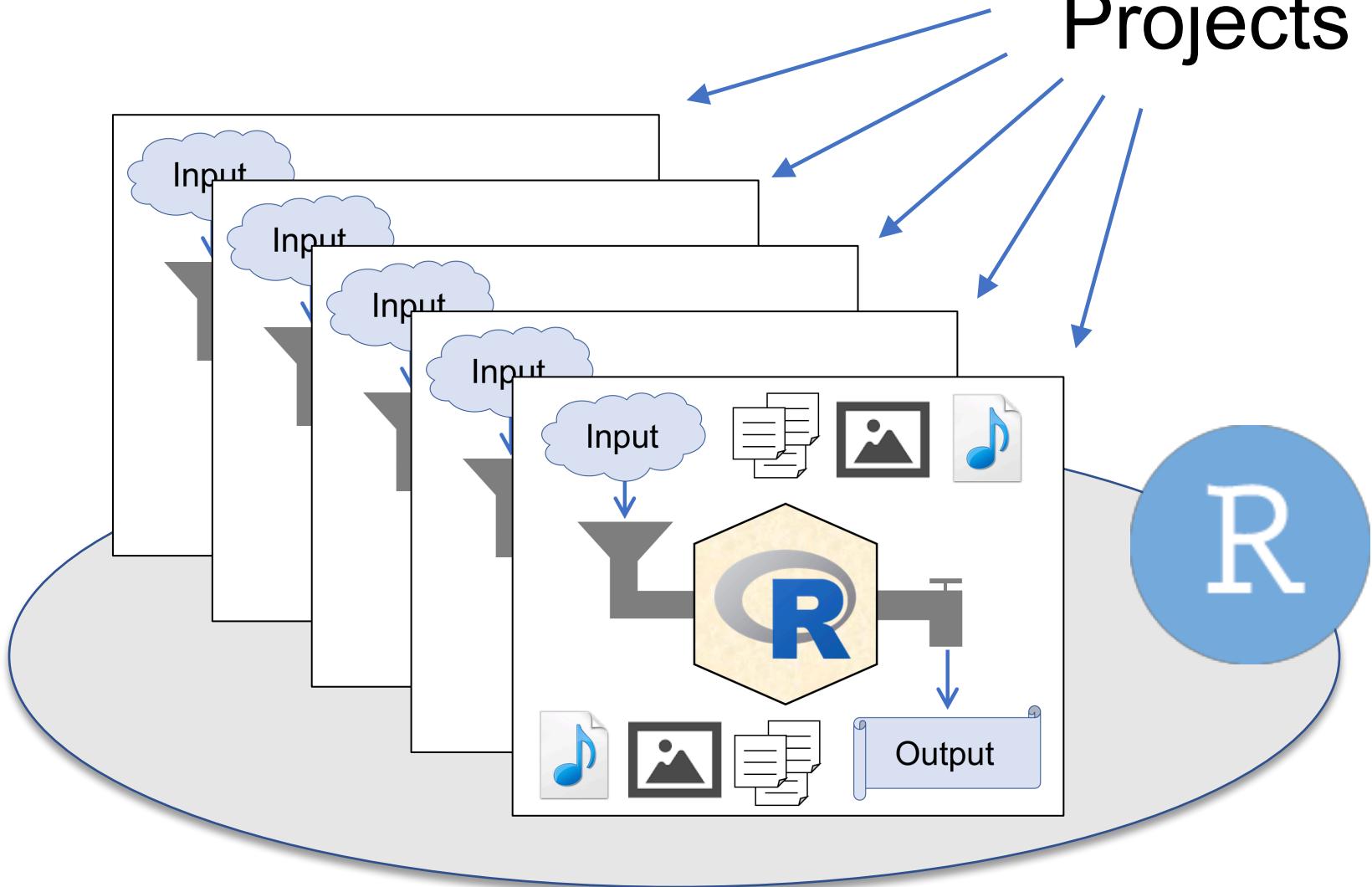


# RStudio

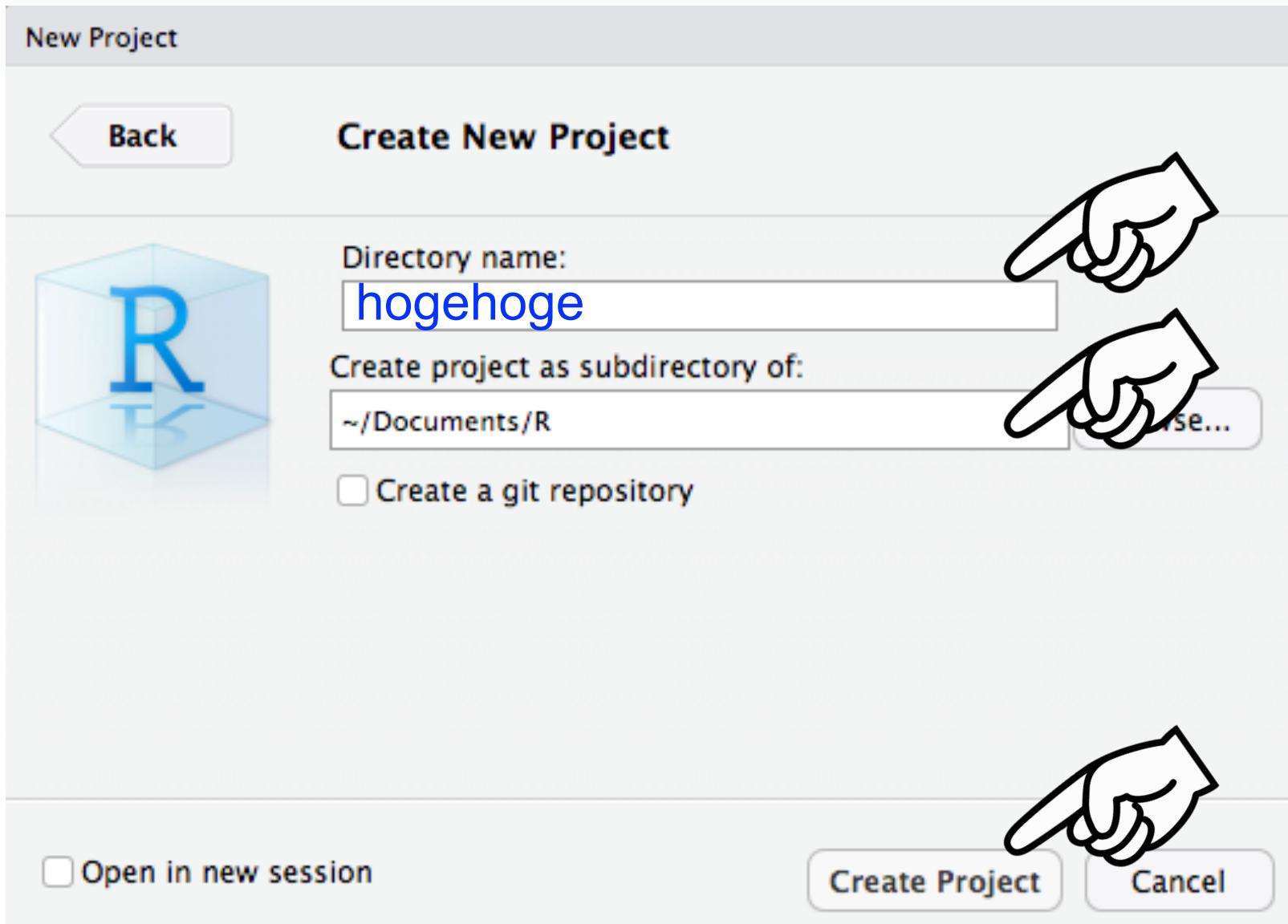


# RStudio

# Projects

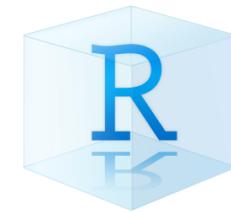


File > New Project... > New Directory > New Project





~/Documents/R



hogehoge.Rproj



.Rproj.user

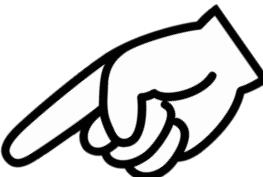


.RData



.Rhistory

# Project Root Directory



Double click!!



Open project



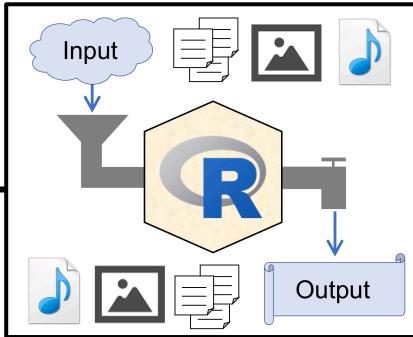
Auto saved  
project information



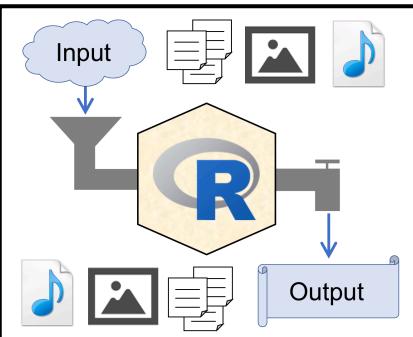
~/Documents/R



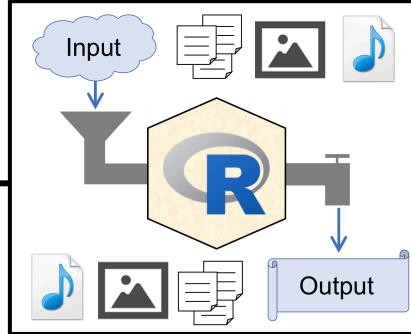
project1



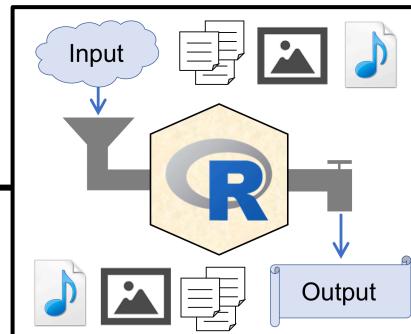
project2



project3



project4





python<sup>TM</sup>

# Variable assignment

var = 1



var <- 1

1 -> var



var = 1



# Variable naming

**1var = 1**

**\_var = 1**

**.var = 1**

**var.1 = 1**

**list = 1**

**1var <- 1**

**\_var <- 1**

**.var <- 1**

**var.1 <- 1**

**list <- 1**



# Variable naming



**1var = 1**



**\_var = 1**



**.var = 1**



**var.1 = 1**



**list = 1**

(reserved)



**1var <- 1**



**\_var <- 1**



**.var <- 1**



**var.1 <- 1**



**list <- 1**



(reserved)



# Variable type

String

```
var <- "1"
```

Integer

```
var <- 1
```

Float

```
var <- 1.0
```

Character

```
var <- "1"
```

Integer

```
var <- 1L
```

Double

```
var <- 1
```

```
var <- 1.0
```



# Variable type

String

```
var <- "1"
```

Integer

```
var <- 1
```

Float

```
var <- 1.0
```

Character

```
var <- "1"
```

Integer

```
var <- 1L
```



Double

```
var <- 1
```

```
var <- 1.0
```



# Packages



you



packages



# Packages

```
import numpy  
numpy.array([1:3])
```

```
library(dplyr)  
filter(dat, ...)
```



# Packages

```
import numpy  
numpy.array([1:3])
```

```
import numpy as np  
np.array([1:3])
```

```
library(dplyr)  
filter(dat, ...)
```

```
dplyr::filter(dat, ...)
```



# Packages

```
import numpy  
numpy.array([1:3])
```

```
import numpy as np  
np.array([1:3])
```

```
from numpy import array  
array([1:3])
```

```
library(dplyr)  
filter(dat, ...)
```

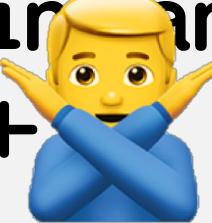
```
dplyr::filter(dat, ...)
```



# Loop

```
for i in range(10):  
    i = i + 1
```

```
for i in range(10):  
    i = i + 1
```



INDENT  
ERROR

```
for i in range(10):  
    i += i
```



```
for(i in 1:10){  
    i = i + 1  
}
```

```
for(i in 1:10)  
    i = i + 1
```



One-liner case



# Function definition

```
def f(x, y = 1):  
    z = x + y  
    return z
```



CHECK YOUR INDENT

```
f <- function(x, y = 1){  
    z = x + y  
    return(z)  
}
```

```
f <- function(x, y = 1){  
    z = x + y  
    # return(z)  
}
```



Autoreturn the final expression (z)



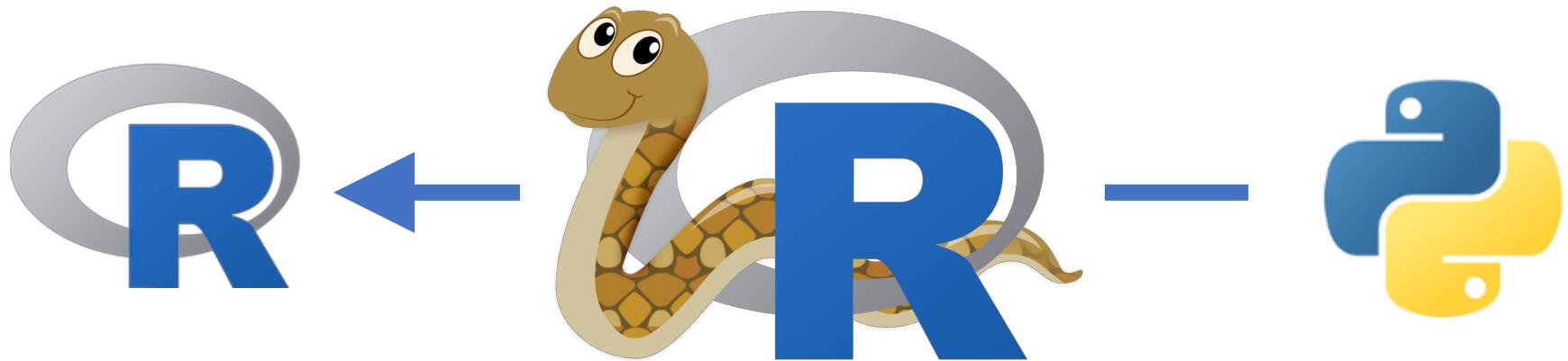




≈



# reticulate package



URL: <https://github.com/rstudio/reticulate>

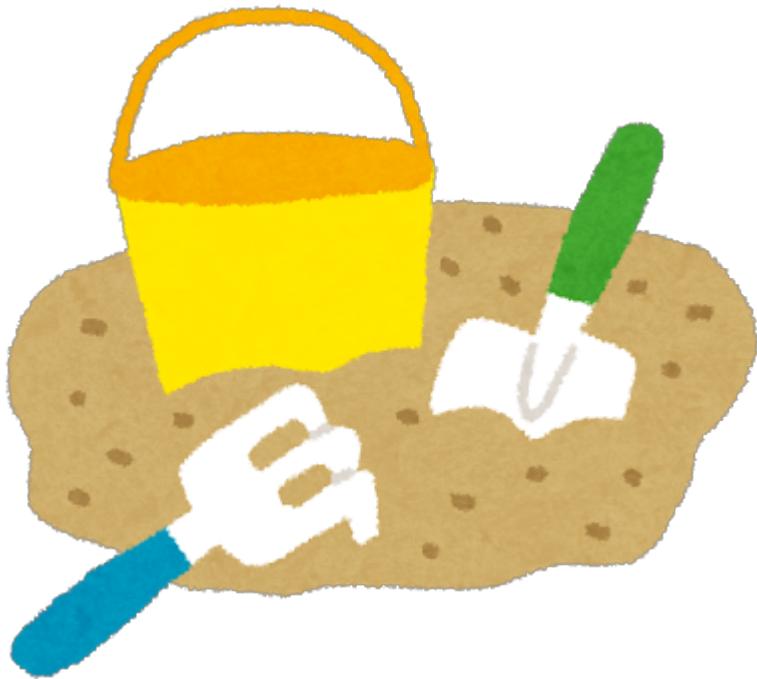
# reticulate package



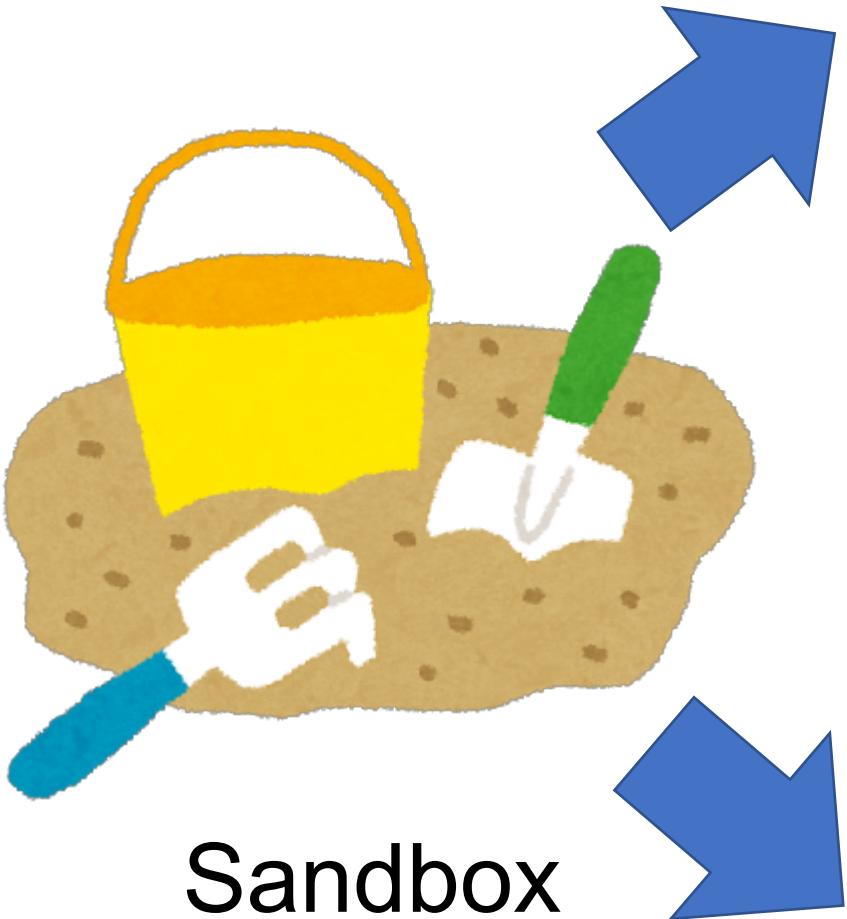
- Calling Python from R  
in a variety of ways including R Markdown,  
sourcing Python scripts,  
importing Python modules,  
and using Python interactively within an R session.
- Translation between R and Python objects  
(for example, between R and Pandas data frames,  
or between R matrices and NumPy arrays).

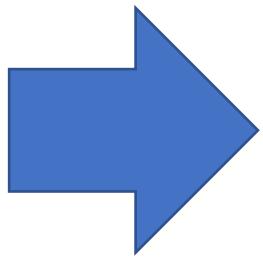
# **Environment setup for Python**

**(in macOS Mojave 10.14.3)**

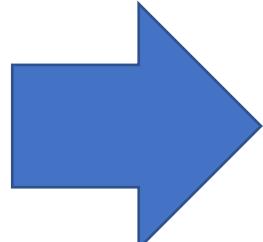
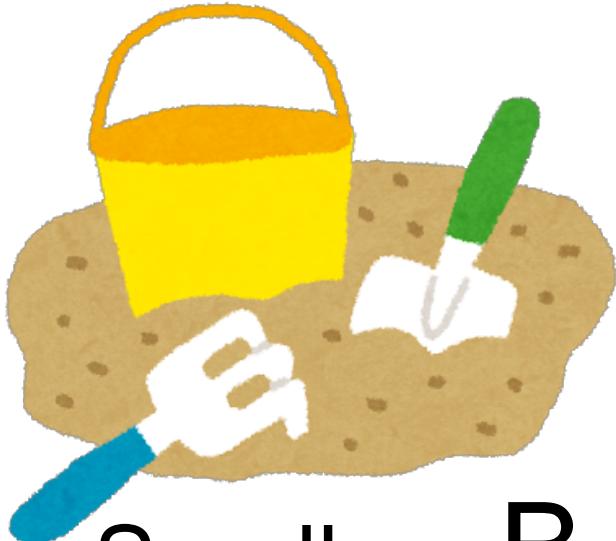


# Sandbox





## Sandbox A



## Sandbox B



<http://buzz-plus.com/2014/07/25/suna/>

Guy-Olivier Deveau sculpture - godeveau@hotmail.com



<http://www.sandart-j.com/work/work3.html>

SandArt Japan



Isolated  
&  
Independent



Sandbox B



<http://buzz-plus.com/2014/07/25/suna/>

Guy-Olivier Deveau sculpture - godeveau@hotmail.com

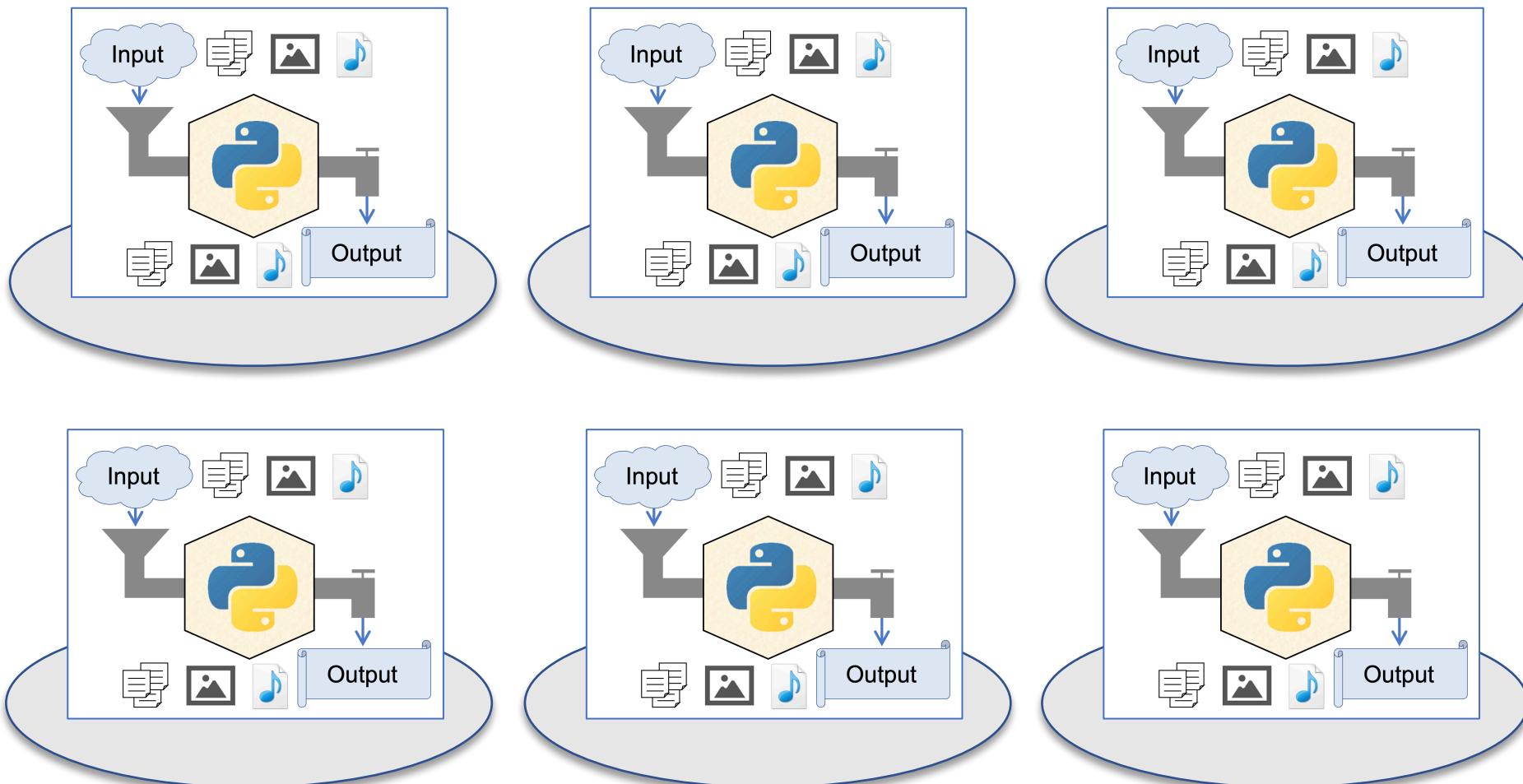


<http://www.sandart-j.com/work/work3.html>

SandArt Japan

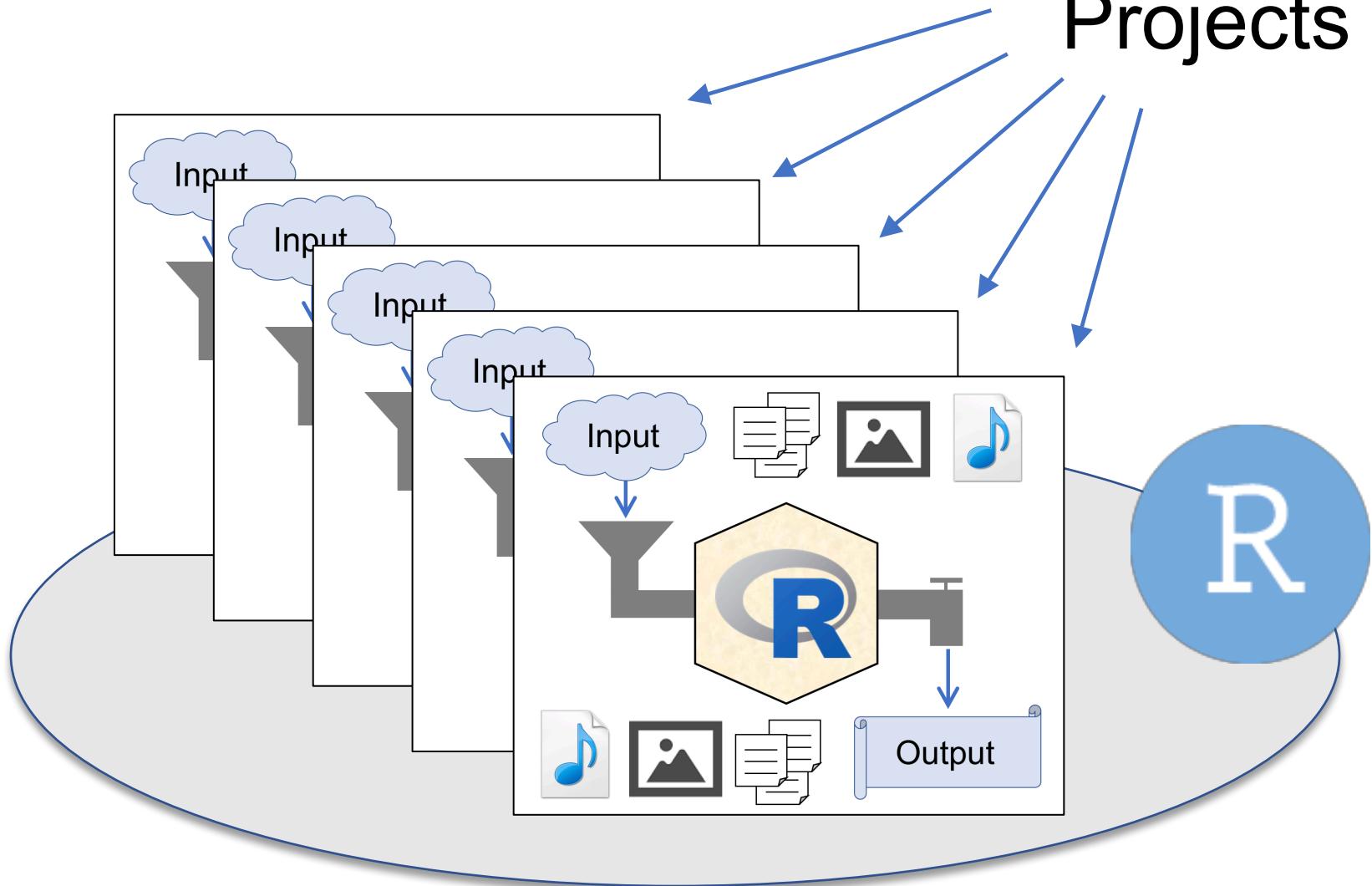
# "Sandboxed" Python

## Isolated & Independent virtual environment for secure and reproducibility



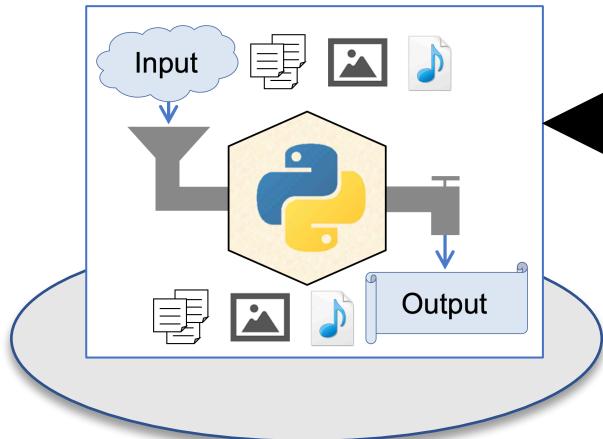
# RStudio

# Projects

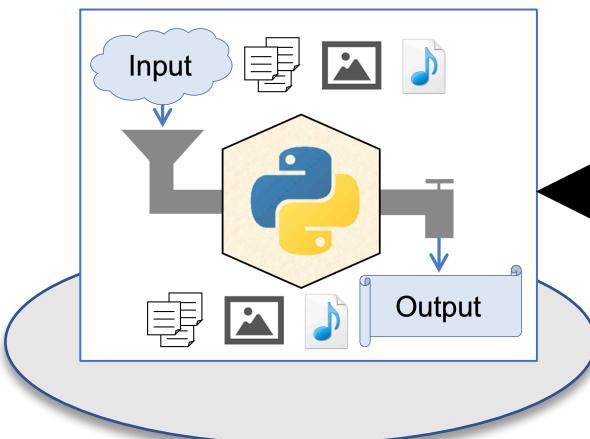


# "Sandboxed" Python

Isolated & Independent virtual environment  
for security & reproducibility



```
[python]  
version = "3.7"  
  
[packages]  
cycler==0.10.0  
kiwisolver==1.1.0  
matplotlib==3.1.1  
numpy==1.16.4
```



```
[python]  
version = "2.7"  
  
[packages]  
numpy==1.16.4  
  
- python==4.1.0.25  
- ==0.25.0  
- ng==2.4.0  
- ==1.1.2  
  
...
```

# Pipenv

→ "Sandboxed" Python manager

## Install Python

<https://www.python.org/>

*Recommended!!*

## Install Pipenv (in MacOS)

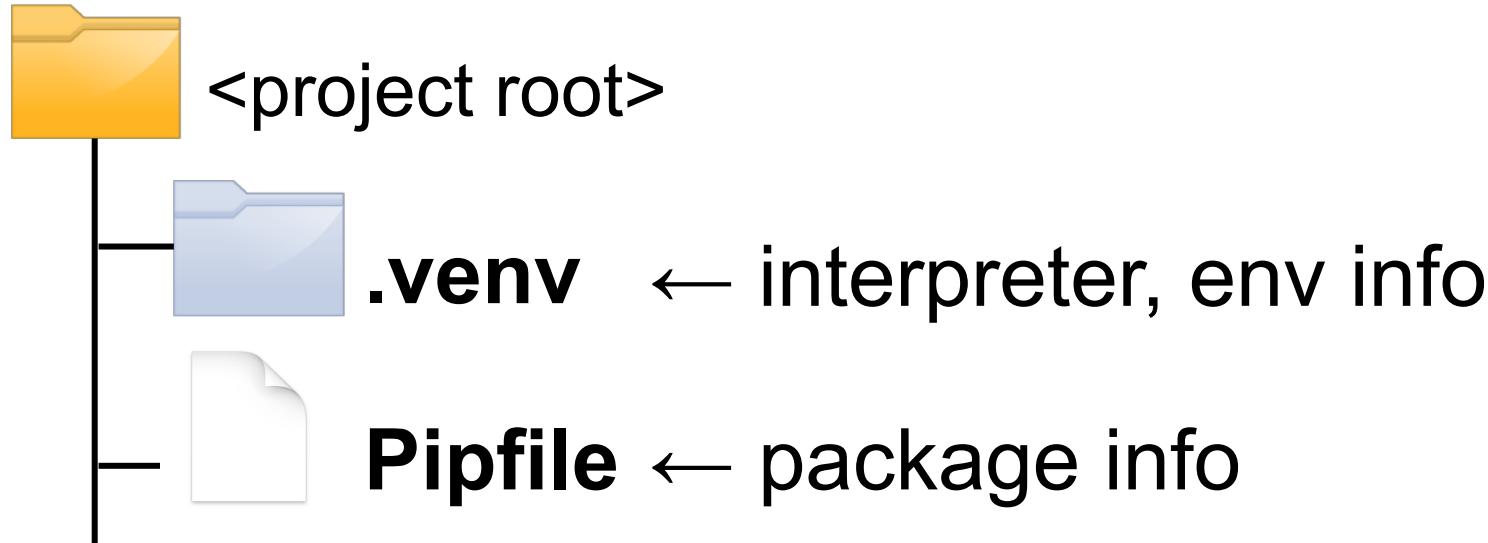
```
$ brew install pipenv
```

# Pipenv

→ "Sandboxed" Python manager

## Create virtualenv

```
$ cd <project root>  
$ pipenv --python 3.7
```



# Pipenv

→ "Sandboxed" Python manager

## Activate virtualenv

```
$ pipenv shell
```

## Deactivate virtualenv

```
(prj) $ exit
```

## Delete virtualenv

```
$ pipenv --rm
```

# Pipenv

→ "Sandboxed" Python manager

## Activate virtualenv

```
$ pipenv shell
```

## Install packages

```
(prj) $ pipenv install <pkg>~=<version>
```

## Uninstall packages

```
(prj) $ pipenv uninstall <pkg>
```

For `{reticulate}`,

you need NumPy package

in your `virtualenv`.

# Pipenv

# → "Sandboxed" Python manager

# Install NumPy

```
$ cd <prj>  
$ pipenv shell # activate  
(prj) $ pipenv install numpy # install
```

```
(prj) $ pipenv run pip freeze # check
```

```
(prj) $ python # check  
=>>> import numpy
```

# Pipenv

→ "Sandboxed" Python manager

## Address of the virtualenv

```
$ cd <prj>
```

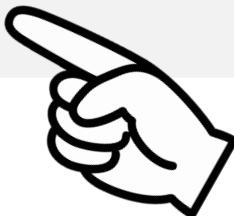
```
$ pipenv shell
```

```
(prj) $ pipenv --venv
```

```
<prj>/venv
```

# activate

# check



back to The logo consists of a large blue capital letter 'R' positioned inside a grey circle. The 'R' is bold and has a slight shadow or glow effect. The circle is also grey and has a thin white outline.

# Use Python in R



Install reticulate from CRAN

```
install.packages(reticulate)
```

Attach Python virtualenv

```
library(reticulate)
```

```
pyenv <- "<prj>/.venv/bin/python"
```

```
use_python(python = pyenv,  
          required = TRUE)
```



# Use Python in R

## Check your Python

### `py_config()`

```
## python: <prj>/.venv/bin/python
## libpython: /Library/Frameworks/Python.framework...
## pythonhome: /Library/Frameworks/Python.fram...
## virtualenv: <prj>/.venv/bin/activate_this.py
## version: 3.7.4 (v3.7.4:e09359112e, Jul 8 2019...
## numpy: <prj>/.venv/lib/python3.7/site-packages...
## numpy_version: 1.16.4
##
## NOTE: Python version was forced by use_python...
```



# Use Python in R

## Import Python pkg in R

```
os <- import("os")
```

## Use Python pkg in R

```
os$listdir()
```

```
## [1] ".Rhistory"          ".DS_Store"  
## [2] ".gitignore"          ".RData"  
## ...
```

# Use Python in R



## Import Python source file

```
source_python("sample.py")
```

```
import pandas as pd

def pd_load_csv(path):
    df = pd.read_csv(path)
    return df

def pd_head(df, n = 3):
    return df.head(n)
```

sample.py



# Use Python in R

## Import Python source

```
source_python("sample.py")
```

```
dat <- pd_load_csv("hoge.csv")
```

```
pd_head(dat)
```

```
iris %>% pd_head
```

# Benchmark test



```
source_python("sample.py")
```

```
f_pd <- function(path) pd_load_csv(path)
```

```
f_base <- function(path) read.csv(path)
```

```
f_readr <- function(path) readr::read_csv(path)
```

```
f_fread <- function(path) data.table::fread(path)
```

# Benchmark test



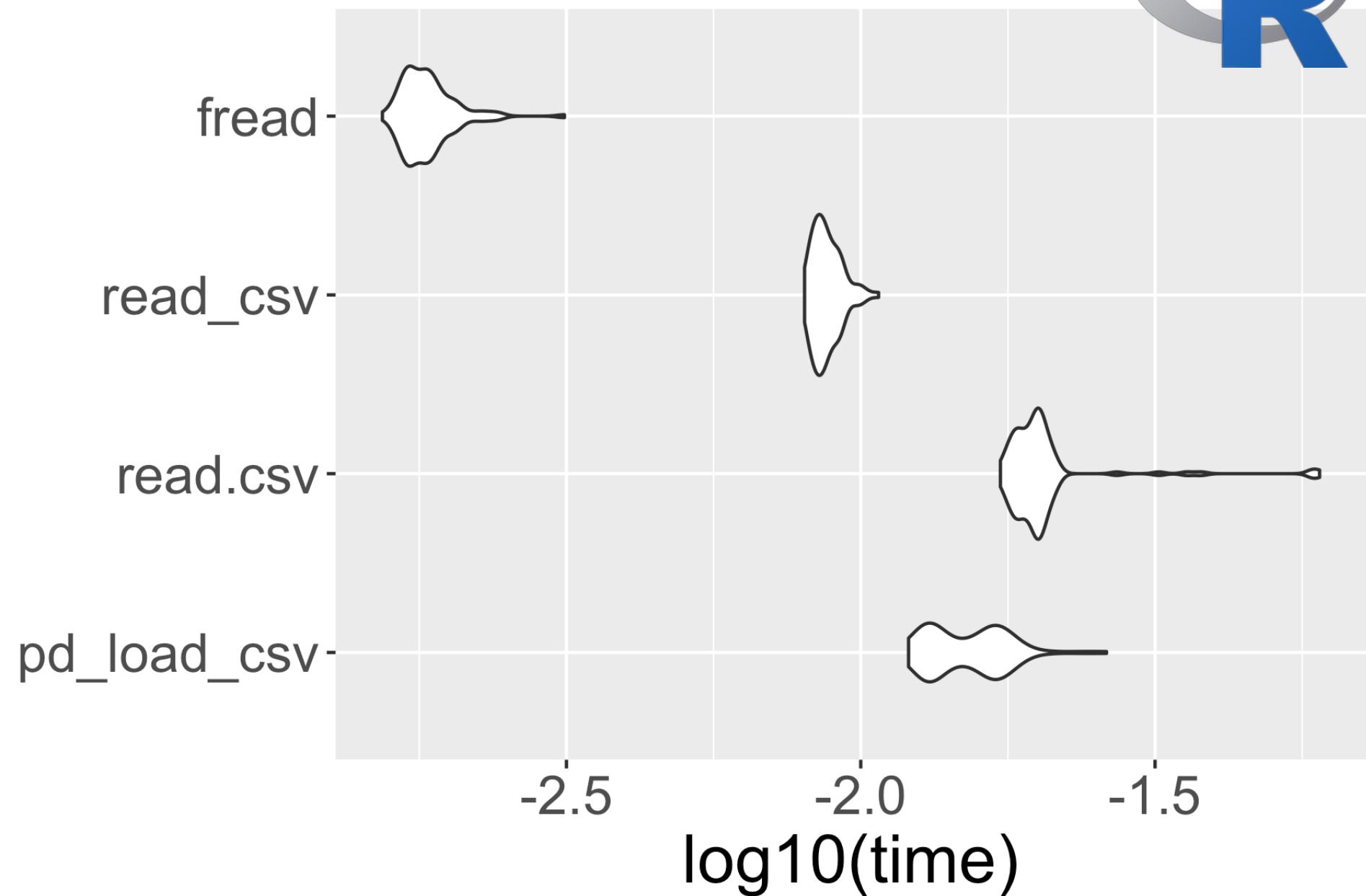
```
source_python("sample.py")
```

```
f_pd <- function(path) pd_load_csv(path)
f_base <- function(path) read.csv(path)
f_readr <- function(path) readr::read_csv(path)
f_fread <- function(path) data.table::fread(path)
```

```
microbenchmark::microbenchmark(
  pd_load_csv = f_pd(path),
  read.csv = f_base(path),
  read_csv = f_readr(path),
  fread = f_fread(path)) -> mbm
```

```
ggplot2::autoplot(mbm)
```

# Benchmark test





# Use Python in R

## Import Python source

```
source_python("sample.py")
```

```
iris %>%  
  pd_head(5)
```

```
## y_call_impl(callable, dots$args, dots$keywords)  
## でエラー:  
## TypeError: cannot do slice indexing on <class  
## 'pandas.core.indexes.range.RangeIndex'> with  
## these indexers [5.0] of <class 'float'>
```

# Variable type

String

```
var <- "1"
```

Integer

```
var <- 1
```

Float

```
var <- 1.0
```

Character

```
var <- "1"
```

Integer

```
var <- 1L
```



Double

```
var <- 1
```

```
var <- 1.0
```



# Use Python in R



## Import Python source

```
source_python("sample.py")
```

```
iris %>%  
  pd_head(5)      # type ERROR
```

```
iris %>%  
  pd_head(5L)    # set as integer
```

# reticulate package

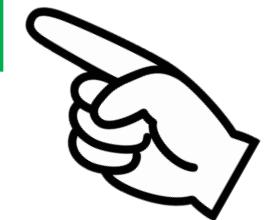


- Calling Python from R  
in a variety of ways including R Markdown,  
**sourcing Python scripts,**  
**importing Python modules,**  
and using Python interactively within an R session.
- Translation between R and Python objects  
(for example, between R and Pandas data frames,  
or between R matrices and NumPy arrays).

# reticulate package



- Calling Python from R  
in a variety of ways including **R Markdown**  
**sourcing Python scripts,**  
**importing Python modules,**  
and using Python interactively within an R session.
- Translation between R and Python objects  
(for example, between R and Pandas data frames,  
or between R matrices and NumPy arrays).



# Use Python in Rmd



## Create .Rmd file

File > New File > R markdown

## Import Python virtualenv in R chunk

```
```{r}
library(reticulate)
pyenv <- "<prj>/.venv/bin/python"
use_python(python = pyenv,
           required = TRUE)
````
```



# Use Python in Rmd

## Use Python in python chunk

```
```{python}
import pandas as pd
path = "<path>/sample.csv"
df = pd.read_csv(path)
df.head(3)
```
```



# Use Python in Rmd

## Use Python in python chunk

```
```{python}
```

```
import pandas as pd
```

```
path = "<path>/sample.csv"
```

```
df = pd.read_csv(path)
```

```
df.head(3)
```

```
```
```



preview



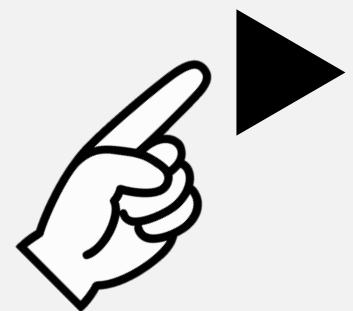


# Use Python in Rmd

## Shear pyobj between pychunks

```
```{python}
import pandas as pd
path = "<path>/sample.csv"
df = pd.read_csv(path)
````
```

```
```{python}
df.head(3)
````
```

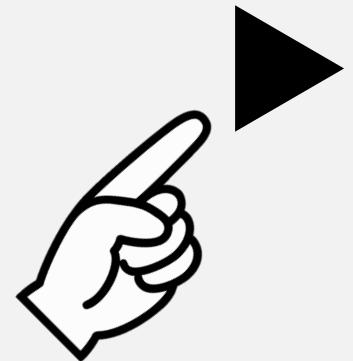




# Use Python in Rmd

## Import R object to python chunk

```
```{python}
import pandas as pd
df = r.iris
df.head(3)
```
```





# Use Python in Rmd

## Import R object to python chunk

```
```{python}
import pandas as pd
df = r.iris
````
```

## Import python object to R chunk

```
```{r}
py <- import_main()
py$df
````
```

# Benchmark test in python chunk

```
```{python}
import pandas as pd
from time import time
path = "<path>/sample.csv"

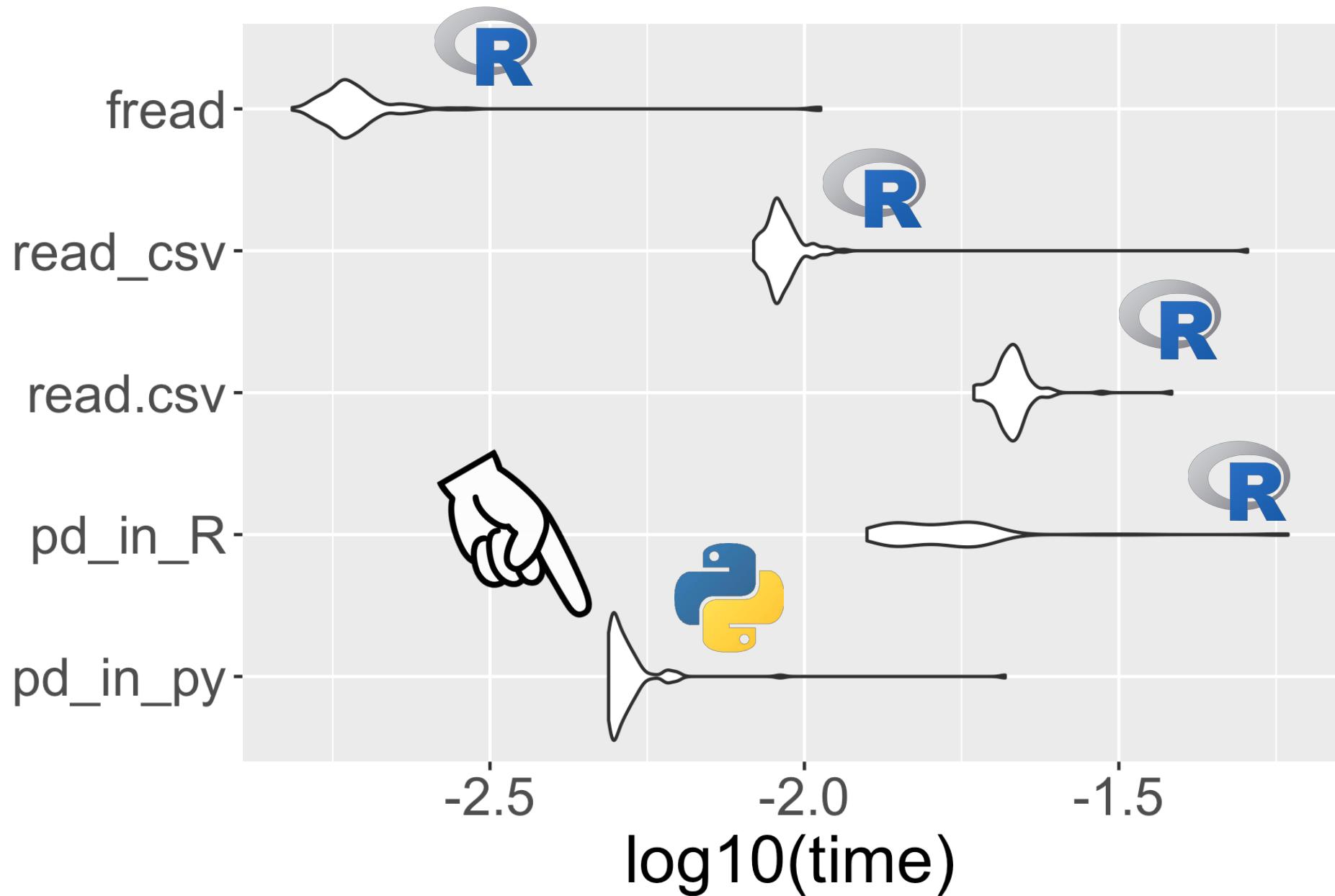
result = []
for i in range(100):
    start = time()
    df = pd.read_csv(path)
    time_i = time() - start
    result.append(time_i)
````
```

# Benchmark visualization in R chunk

```
```{r}
py <- import_main()

py$result %>%
  data.frame(expr = "py_pd", time = .) %>%
  rbind(data.frame(mbm) %>%
    mutate(time = time/10^9)) %>%
  ggplot(aes(expr, log10(time)))+
  geom_violin()+
  coord_flip()
```
```

# Benchmark visualization in R chunk



# reticulate package



- Calling Python from R
  - in a variety of ways including R Markdown,  
sourcing Python scripts,  
importing Python modules,  
and using Python interactively within an R session.

# reticulate package



- Calling Python from R

- in a variety of ways including **R Markdown**,  
**sourcing Python scripts**,  
**importing Python modules**.



- and **using Python interactively within an R session**.



# Run Python on Rstudio



## 1. Attach Python virtualenv in R

```
library(reticulate)  
pyenv <- "<prj>/.venv/bin/python"  
use_python(python = pyenv,  
           required = TRUE)
```



## 2. Create .py file

File > New File > Python script

## 3. write in .py file

```
a = 1
```





```
> reticulate::repl_python()
Python 3.7.4
Reticulate 1.12 REPL -- A Pyt
>>> a = 1
>>> |
```

```
> reticulate::repl_python()
Python 3.7.4
Reticulate 1.12 REPL -- A Pyt
>>> a = 1
>>> import pandas as pd
>>> df = r.iris
>>> df.head(3)
   Sepal.Length  Sepal.Width  Petal.Length  Petal.Width
0            5.1          3.5          1.4          0.2
1            4.9          3.0          1.4          0.3
```

```
> reticulate::repl_python()
Python 3.7.4
Reticulate 1.12 REPL -- A Pyt
>>> import pandas as pd
>>> df = r.iris
>>> (escape key)
> import_main()$df %>% head
  Sepal.Length Sepal.Width Petal.
1          5.1         3.5
2          4.9         3.0
3          4.7         3.2
```

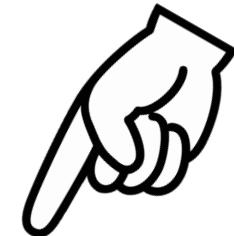
```
> reticulate::repl_python( )
>>> import pyper
>>>
>>> r = pyper.R()
>>> r("set.seed(71)")
'try({set.seed(71)})\n'
>>> r("unif_r <- runif(10, 0, 1)")
'try({unif_r <- runif(10, 0, 1)})\n'
>>>
>>> result = r.get("unif_r")
>>>
>>> result
array([0.33292806, 0.55510387, 0.3273699
```



# reticulate package



- Calling Python from R  
in a variety of ways including  
**R Markdown,**  
**sourcing Python scripts,**  
**importing Python modules.**



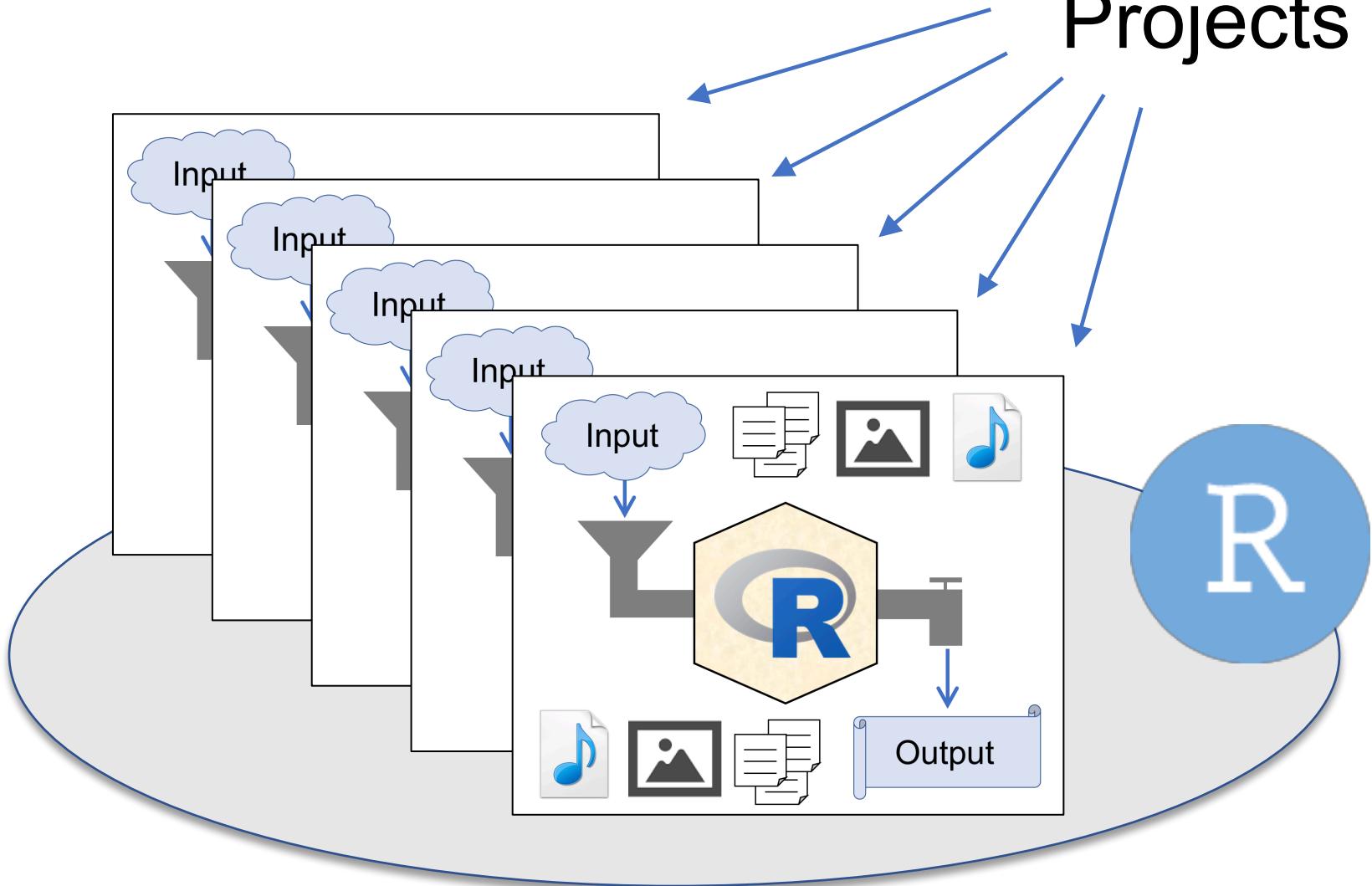
and **using Python interactively within an R session.**



summary

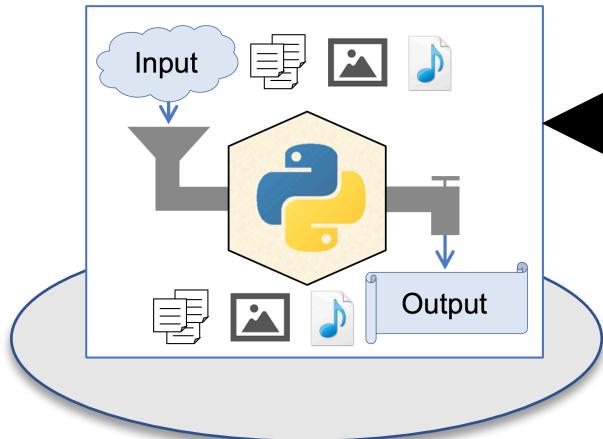
# RStudio

# Projects

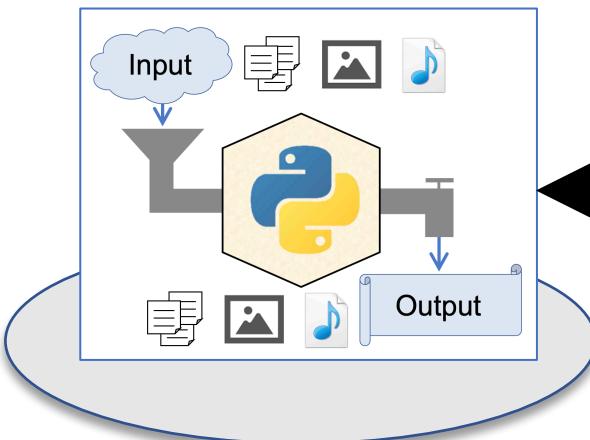


# "Sandboxed" Python

Isolated & Independent virtual environment  
for security & reproducibility



```
[python]  
version = "3.7"  
  
[packages]  
cycler==0.10.0  
kiwisolver==1.1.0  
matplotlib==3.1.1  
numpy==1.16.4
```



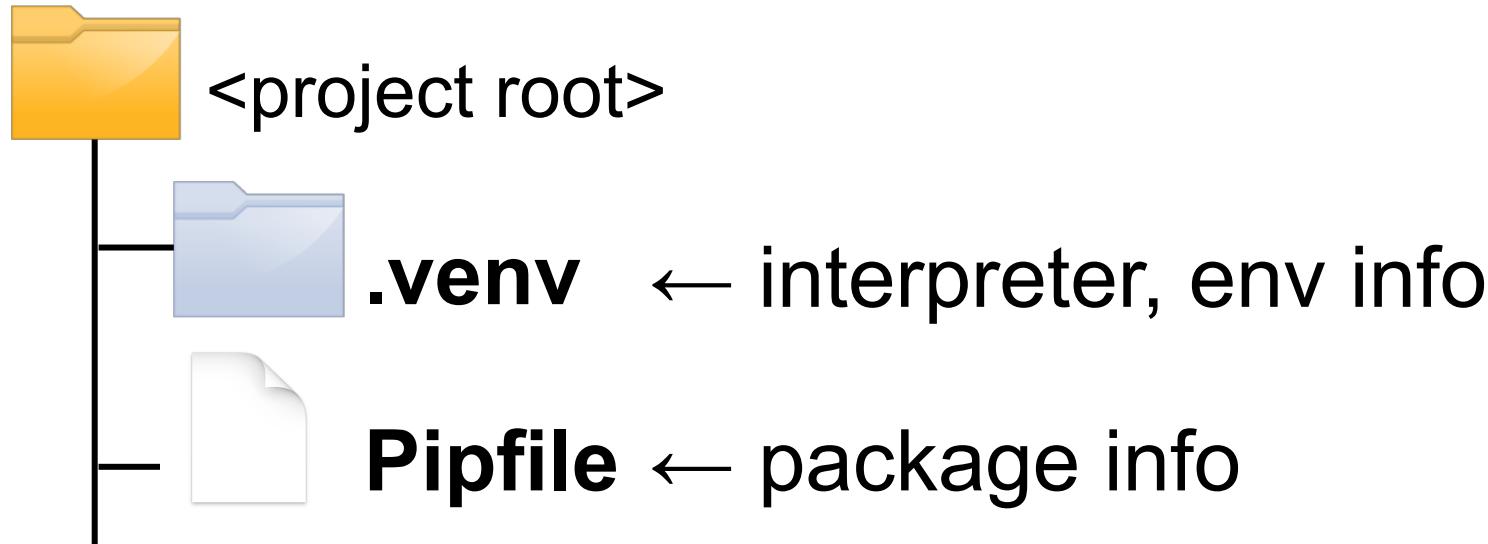
```
[python]  
version = "2.7"  
  
[packages]  
numpy==1.16.4  
  
- python==4.1.0.25  
- ==0.25.0  
- ng==2.4.0  
- ==1.1.2  
  
...
```

# Pipenv

→ "Sandboxed" Python manager

## Create virtualenv

```
$ cd <project root>  
$ pipenv --python 3.7
```



# Use Python in R



Install reticulate from CRAN

```
install.packages(reticulate)
```

Attach Python virtualenv

```
library(reticulate)
```

```
pyenv <- "<prj>/.venv/bin/python"
```

```
use_python(python = pyenv,  
          required = TRUE)
```



# Use Python in R

## Import Python source

```
pd <- import("pandas")
source_python("sample.py")
```

## Shear pyobj between pychunks in Rmd

```
```{python}
import pandas as pd
path = "<path>/sample.csv"
df = pd.read_csv(path)
````
```



# Run Python in Rstudio



## 1. Attach Python virtualenv in R

```
library(reticulate)  
pyenv <- "<prj>/.venv/bin/python"  
use_python(python = pyenv,  
           required = TRUE)
```



## 2. Create .py file

File > New File > Python script

## 3. write in .py file

```
a = 1
```



# reticulate package



- Calling Python from R  
in a variety of ways including  
R Markdown,  
sourcing Python scripts,  
importing Python modules,  
and using Python interactively within an R session.

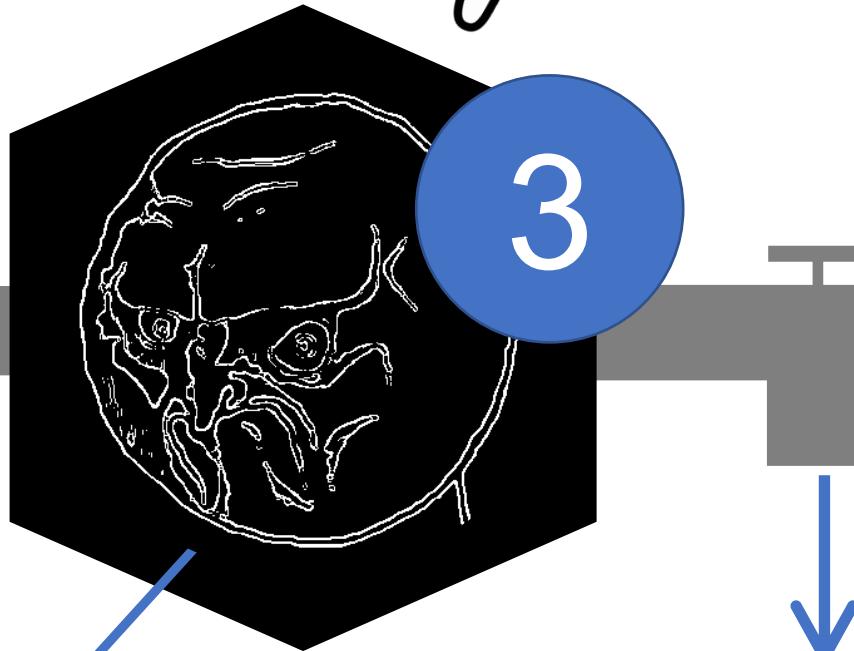


**Input**

2



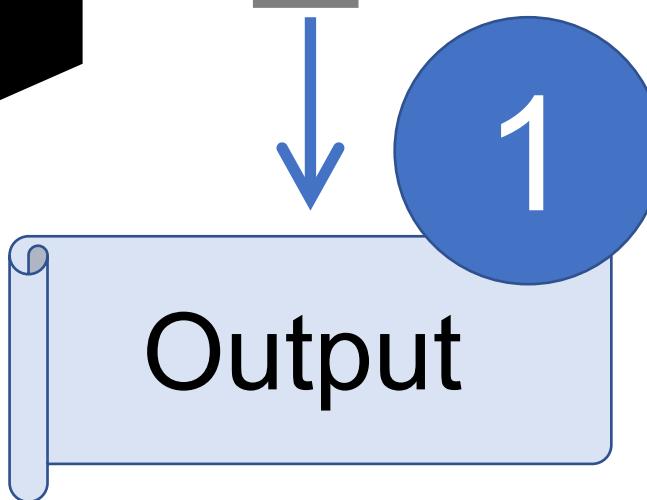
Do NOT start from here



Whatever



1



Output



