Run Bash Script in R and Python

Yingqi Jing

March 10, 2020

Contents

List of Tables

List of Figures

S1 R scripts

In order to run the bash command, we need to use the `system()` function in R. We first need to write the command in a string or a vector of characters, and feed the command in the system function. For example, we can list and filter all the pdfs in my Documents.

```
bash_command_r = paste("ls -a ~/Documents/", "|", "grep", "'pdf'")
try(system(bash_command_r, intern = TRUE, ignore.stderr = F, ignore.stdout = F))
```

```
[1] "2018_06_20_Wednesday_09h00-Birth-death.pdf" "Virus test.pdf"                                "newpaymer
```

S2 Python scripts

In python, you need to use the `subprocess` module to run the bash commands. Of course, there is also a `os.system()` function, which can run simple command directly. For more advance usages, it is better to use the `subprocess` module.

```
import subprocess
cmd = "ls -a ~/Documents/|grep 'pdf'"
# Note: careful about the use of shell=True (strongly discouraged)
ps = subprocess.Popen(cmd, shell=True,stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
output = ps.communicate()[0]
print(output)
```

```
b'2018_06_20_Wednesday_09h00-Birth-death.pdf\nVirus test.pdf\nnewpayment.pdf\n'
```

```
# Popen() function will create an instance
# where communicate() function will return the standard output and error
c = subprocess.Popen(['ls','-a','/Users/jakejing/Documents/'], stdout = subprocess.PIPE, stderr = subpr
stdout, stderr = c.communicate()

print(stdout)
```

```
b".\n..\n.DS_Store\n.ipynb_checkpoints\n.localized\n.vimrc\n2018_06_20_Wednesday_09h00-Birth-death.pdf\n
```

```
print(stderr)
```

```
b''
```

```
subprocess.check_call(['/Users/jakejing/git/run-bash-R-Python/program.sh',"/Users/jakejing/Documents/"]
```

```
0
```

S3 References