Scheduling jobs (RStudio, Shell)

Date: Wednesday, June 3rd, 2020
By: Viviana Márquez | Data Scientist/Data Steward

R Studio

Fortunately, R Studio has a plug-in that makes scheduling jobs easier than what it is in Python.

taskscheduleR

Works on Windows, more info at: https://github.com/bnosac/taskscheduleR

cronR

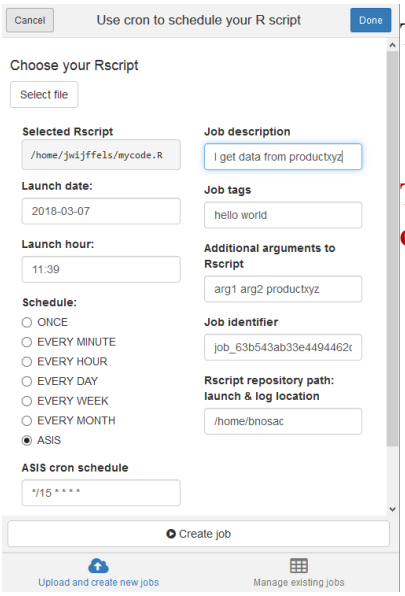
Works on Unix/Linux (works on the AMI).
More info at: https://github.com/bnosac/cronR

To install, go to **sudo R** and install packages:

```
[ec2-user@ip-10-243-100-126 ~]$ sudo R
```

```
install.packages('devtools')
devtools::install_github("bnosac/cronR")

install.packages('miniUI')
install.packages('shiny')
install.packages('shinyFiles')
```



Then, it is as easy as just clicking on the Addins:

TODO: Check if the plug0in overwrites previous crontabs.

Shell

The best way to set up scheduled jobs is by doing it directly on the shell where we have full control.

Useful commands:

- Check current time
```
[ec2-user@ip-10-243-100-126 ~]$ date
```
- Repeat 'Hello World!'
```
[ec2-user@ip-10-243-100-126 ~]$ echo Hello World!
```
- Repeat "Today <current time>"
```
[ec2-user@ip-10-243-100-126 ~]$ echo Today `date`
```
- Save output in a log file. (> overwrites, >> appends)
```
[ec2-user@ip-10-243-100-126 ~]$ echo Today `date` >> output.log
```
- Run a Python file
```
[ec2-user@ip-10-243-100-126 ~]$ python shell_demo.py
```
- See all created jobs
```
[ec2-user@ip-10-243-100-126 ~]$ crontab -l
```
- Add a new job or edit an existing one
```
[ec2-user@ip-10-243-100-126 ~]$ crontab -e
```
This will open a **Vi** editor. In this file, each line is a job.

Vi editor basic commands:

  - **i**
    insert mode

  - **esc**
    exit insert mode

  - **:q!**
    quit without saving changes

  - **:wq**
    quit and save changes

Structure of a cronjob:

```
* * * * * <command-to-excecute>
- - - - -
| | | | |
| | | | ----- Day of week (0 - 7) (Sunday=0 or 7)
| | | ------- Month (1 - 12)
| | --------- Day of month (1 - 31)
| ----------- Hour (0 - 23)
------------- Minute (0 - 59)
```

Some examples:

- Run job every minute
  ```
  * * * * * <command-to-excecute> # comment
  ```
- Run job every five minutes
  ```
  */5 * * * * <command-to-excecute>
  ```
- Run job every hour at minute 5 (i.e. 10:05, 11:05, 12:05, etc.)
  ```
  5 * * * * <command-to-excecute>
  ```
- Run job every Monday in April at 3:05
  ```
  5 3 * 4 1 <command-to-excecute>
  ```
- Run a Python job every minute and save output (it's always a good practice to save log files).
  ```
  * * * * * python shell_demo.py >> output.log
  ```

Helper: https://crontab.guru/

Shell script:

The best way to create a job, is to write a shell script that takes care of all issues (which python, path location, etc.) You can create a new shell script using Vi.

```
#!/usr/bin/bash

source activate python3
python --version
cd $HOME/Viviana/Demos
ehco Location `pwd`
python shell_demo.py
echo Done!
```

To run:

```
[ec2-user@ip-10-243-100-126 ~]$ chmod +x shell_demo.sh
[ec2-user@ip-10-243-100-126 ~]$ ./shell_demo.sh
```

We can also set the shell script as a scheduled job.