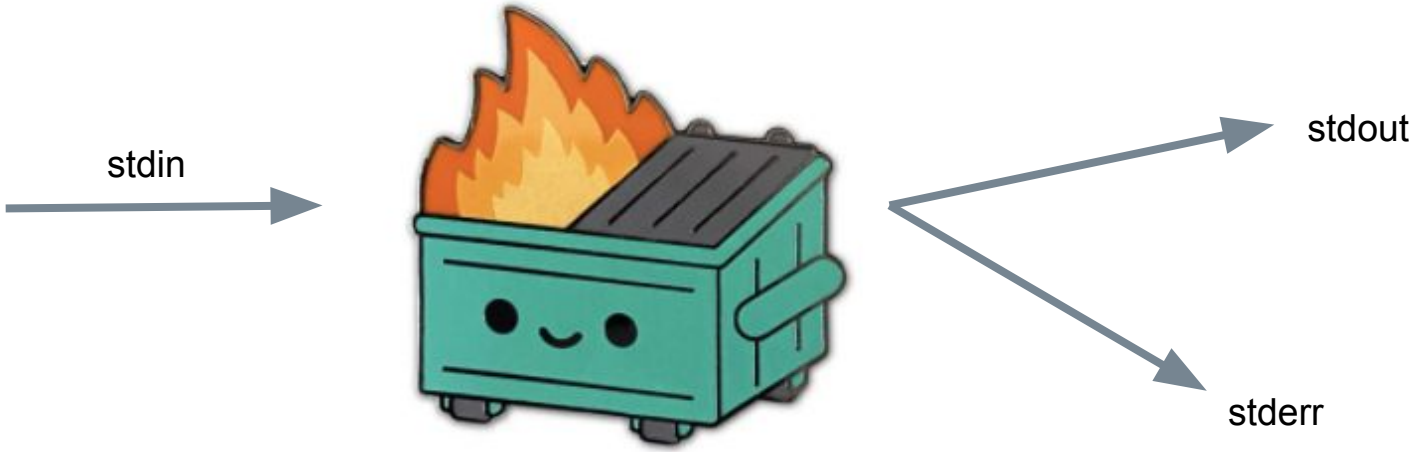
R From the Command Line

 /jameslamb  @_jameslamb

Command Line Concepts

Processes take input from stdin and optionally produce output to stdout and / or stderr.



A process's output streams can be directed to a console, a file, or other process.

```
./r-from-the-command-line $ cat < README.md | grep -E '^#' > out.txt
./r-from-the-command-line $ cat out.txt
# r-from-the-command-line
## Demo Code
## Where this talk has been given
```

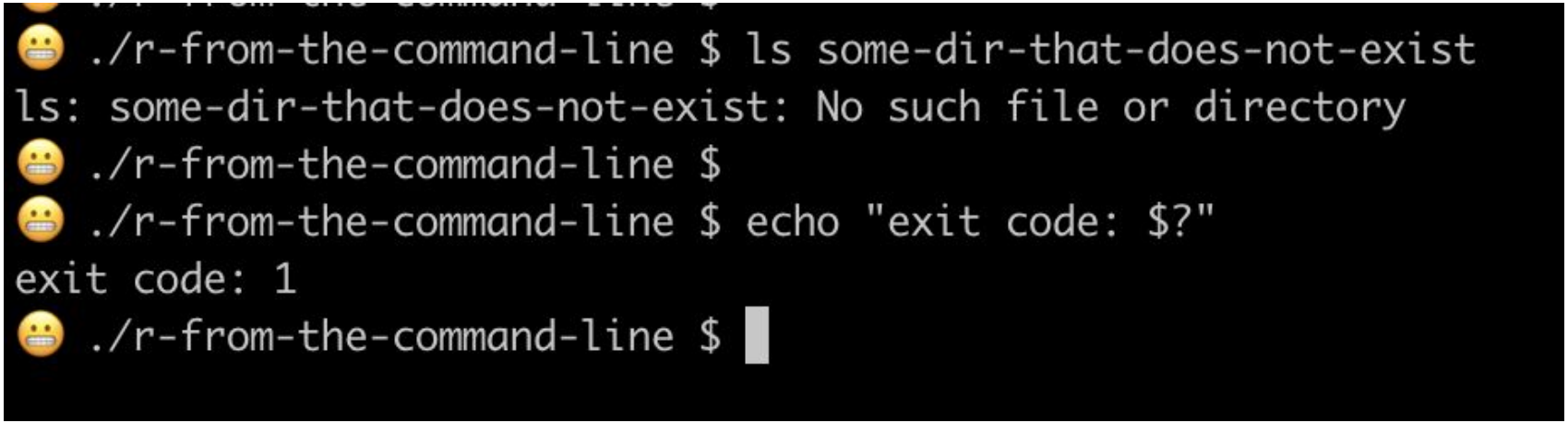
Command-line Tools can be 235x Faster than your Hadoop Cluster

January 18, 2014

Share this: twitter // facebook // linkedin // google+

los angeles

Processes return an integer status called an “exit code”. Anything other than 0 means the program failed...sort of.



Continuous Integration (CI) tools treat any step which returns a non-0 exit code as “failed”. That’s why so many testing frameworks return a non-0 code if your tests fail!

exit code = 0

exit code > 0

Intro to Rscript

Rscript can be used to run code in an R script.

```
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $ Rscript sample-r-code/print-random-numbers.R
[1] "I am printing random numbers"
[1] -1.503931
[1] 1.414989
[1] 0.1298282
[1] -0.4142262
[1] -1.174137
[1] -0.7255627
[1] -0.3647678
[1] -0.1240298
[1] -1.182944
[1] -0.07303144
[1] "done printing numbers"
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $
```

Why use this instead of just opening RStudio?

- Works in environments without a GUI
  - continuous integration
  - configuring servers
  - building containers
- Can be chained with other programs
- Output can be redirected to file easily

(--e) run R code passed in a string

```
😀 ./r-from-the-command-line $ Rscript -e "library(data.table); data.table(m = rnorm(10))"
             m
 1: -1.2465082
 2:  0.4042522
 3: -1.3301165
 4:  0.2692688
 5: -2.1856099
 6: -1.5226256
 7: -0.6297068
 8: -0.3821846
 9: -1.3833027
10: -1.0074923
😀 ./r-from-the-command-line $
```

print() goes to stdout, message(), warning(), stop() go to stderr

```
😬 ./talks $ Rscript \
>     -e "print('printing'); message('messaging'); warning('warning'); stop('stopping')" \
>     1> stdout.txt \
>     2> stderr.txt
😬 ./talks $
😬 ./talks $ cat stdout.txt
[1] "printing"
😬 ./talks $
😬 ./talks $ cat stderr.txt
messaging
Warning message:
warning
Error: stopping
Execution halted
😬 ./talks $
```

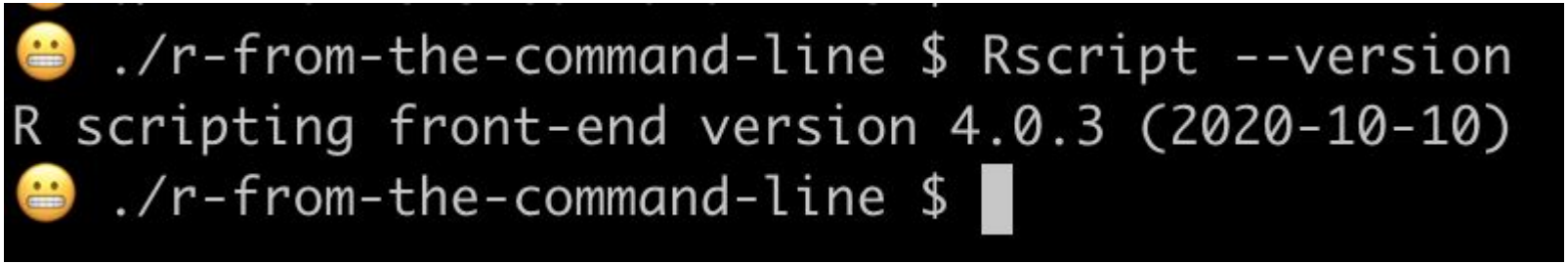
(--help) see Rscript's documentation

```
😬 ./r-from-the-command-line $ Rscript --help
Usage: /path/to/Rscript [--options] [-e expr [-e expr2 ...] | file] [args]

--options accepted are
  --help              Print usage and exit
  --version           Print version and exit
  --verbose           Print information on progress
  --default-packages=list
                      Where 'list' is a comma-separated set
                        of package names, or 'NULL'
or options to R, in addition to --no-echo --no-restore, such as
  --save              Do save workspace at the end of the session
  --no-environ        Don't read the site and user environment files
  --no-site-file      Don't read the site-wide Rprofile
  --no-init-file      Don't read the user R profile
  --restore           Do restore previously saved objects at startup
  --vanilla           Combine --no-save, --no-restore, --no-site-file
                        --no-init-file and --no-environ

'file' may contain spaces but not shell metacharacters
Expressions (one or more '-e <expr>') may be used *instead* of 'file'
See also  ?Rscript  from within R
😬 ./r-from-the-command-line $
```

(--version) see the current version of R



(--verbose) print extra diagnostic information

```
😬 ./r-from-the-command-line $ Rscript --verbose -e "library(data.table);
data.table(m = rnorm(10))"
running
  '/Library/Frameworks/R.framework/Versions/4.0/Resources/bin/R --no-echo
--no-restore -e library(data.table); data.table(m = rnorm(10))'

             m
 1: -0.88005249
 2:  0.38496683
 3: -1.08475672
 4:  0.72495675
 5:  1.31792362
 6: -0.01168519
 7: -1.72214027
 8:  0.52992821
 9: -0.45642013
10:  0.93292442
😬 ./r-from-the-command-line $
```

(--default-packages={list}) load libraries before running any code

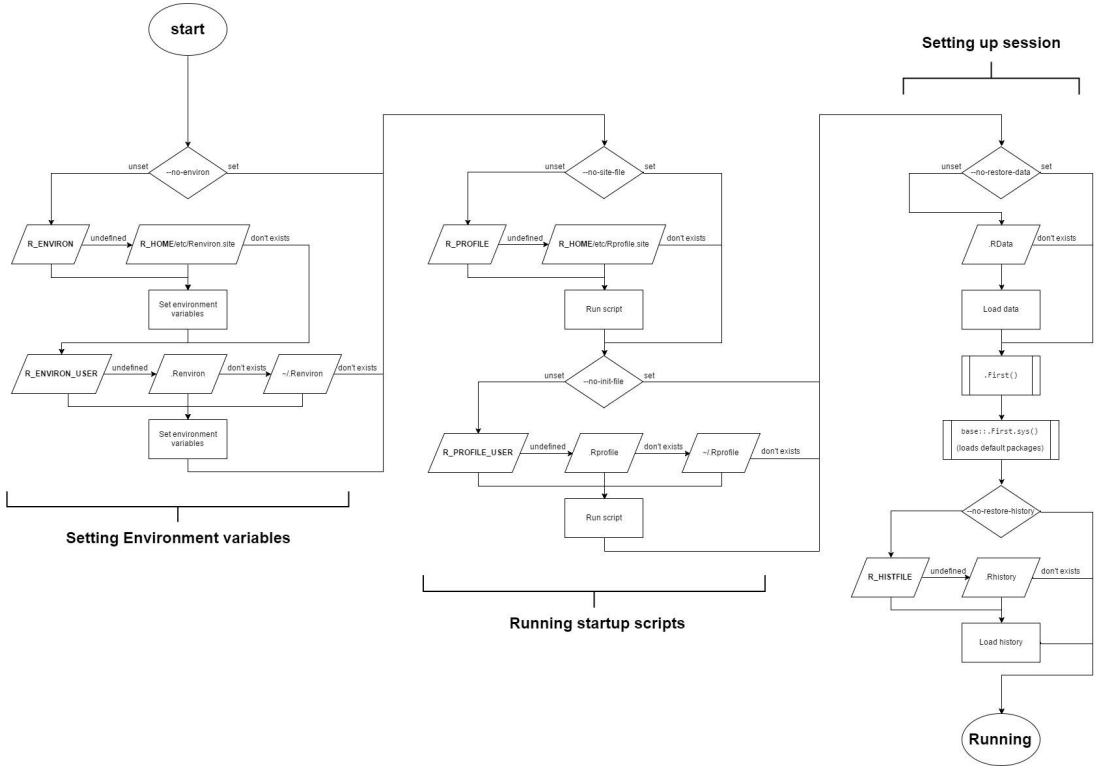
```
😬 ./r-from-the-command-line $ Rscript -e "data.table(m = rnorm(10))"
Error in data.table(m = rnorm(10)) : could not find function "data.table"
Execution halted
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $ Rscript --default-packages="data.table,stats" -e "data.table(m = rnorm(10))"
            m
 1: -1.1439311
 2:  0.1853920
 3: -2.3771550
 4:  0.2617524
 5: -0.2855616
 6: -0.3109145
 7: -0.3957713
 8: -0.7548992
 9: -0.7685032
10: -0.1135844
😬 ./r-from-the-command-line $
```

(--save) save the environment to a .Rdata file before exiting

(--restore / --no-restore) do / do not load .Rdata when running code

```
😬 ./r-from-the-command-line $ Rscript --save -e "some_var <- 11.5"
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $ Rscript --no-restore -e "print(some_var)"
Error in print(some_var) : object 'some_var' not found
Execution halted
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $ Rscript --restore -e "print(some_var)"
[1] 11.5
😬 ./r-from-the-command-line $
😬 ./r-from-the-command-line $
```

(--no-site-file / --no-init-file) ignore different .Rprofile files
(--no-environ) ignore .Renviron files



(--vanilla) don’t load .Rdata or any stuff from .Rprofile / .Renviron

Getting fancy: commandArgs, {argparse}, and {crayon}

Live demo: lint arbitrary R code with {lintr}

```
😬 ./r-from-the-command-line $ Rscript lint-pretty.R --dir-to-lint=$(pwd)/sample-r-code
 ----------------------
| lInTINg Is ImPORtAnT |
 ----------------------
  \\
   \\    *
          *
     ----//-------
     \..C/--..--/ \    `A
      (@ )  ( @) \  \// |w
       \          \  \---/
        HGGGGGGG    \    /`
        V `---------`--'
            <<    <<
           ###   ###


print-random-numbers.R:2:7: style: Only use double-quotes.
print('I am printing random numbers')
      ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
print-random-numbers.R:4:19: style: Integers should not be implicit. Use the form 1L for integers or 1.0 for doubles.
for (n in rnorm(10)){
                  ~~^
print-random-numbers.R:4:20: style: There should be a space between right parenthesis and an opening curly brace.
for (n in rnorm(10)){
                    ^~
print-random-numbers.R:8:7: style: Only use double-quotes.
print('done printing numbers')
      ^~~~~~~~~~~~~~~~~~~~~~

---------- 4 issues found ----------
😬 ./r-from-the-command-line $
```

For a more complicated case, see the main {lightgbm} script

Install with CMake

After following the "preparation" steps above for your operating system, build and install the R-package with the following commands:

```
git clone --recursive https://github.com/microsoft/LightGBM
cd LightGBM
Rscript build_r.R
```

The `build_r.R` script builds the package in a temporary directory called `lightgbm_r`. It will destroy and recreate that directory each time you run the script. That script supports the following command-line options:

- `--skip-install`: Build the package tarball, but do not install it.
- `--use-gpu`: Build a GPU-enabled version of the library.
- `--use-mingw`: Force the use of MinGW toolchain, regardless of R version.
- `--use-msys2`: Force the use of MSYS2 toolchain, regardless of R version.

https://github.com/microsoft/LightGBM/blob/master/build_r.R

There are a few other packages you might find useful for CLIs

{futile.logger} - customizable logger

{jsonlite} - fast, lightweight JSON reader / writer

{optparse} - writes CLIs with help text and command line arguments

{praise} - generates random encouraging messages (nice for the end of a program!)

{processx} - run shell commands from R, with some nice features not available in system()

{progress} - print progress bars for long-running operations

Thanks for your time!

/jameslamb

@_jameslamb

jaylamb20@gmail.com

https://github.com/jameslamb/talks/tree/main/r-from-the-command-line