argparse Command Line Argument Parsing

January 29, 2013

argparse is a command line argument parser inspired by Python's "argparse" library. Use this with Rscript to write "#!"-shebang scripts that accept short and long flags/options and positional arguments, generate a usage statement, and set default values for options that are not specified on the command line.

In our working directory we have two example R scripts, named "example.R" and "display_file.R" illustrating the use of the argparse package.

```
bash$ ls
```

```
display_file.R
example.R
```

In order for a *nix system to recognize a "#!"-shebang line you need to mark the file executable with the "chmod" command, it also helps to add the directory containing your Rscripts to your path:

```
bash$ chmod ug+x display_file.R example.R
bash$ export PATH=$PATH:`pwd`
```

Here is what "example.R" contains:

```
bash$ display_file.R example.R
```

```
#!/usr/bin/env Rscript
# Note:  This example is a port of an example in the getopt package
#        which is Copyright 2008 Allen Day
suppressPackageStartupMessages(library("argparse"))

# create parser object
parser <- ArgumentParser()

# specify our desired options
# by default ArgumentParser will add an help option
parser$add_argument("-v", "--verbose", action="store_true", default=TRUE,
    help="Print extra output [default]")
parser$add_argument("-q", "--quietly", action="store_false",
    dest="verbose", help="Print little output")
parser$add_argument("-c", "--count", type="integer", default=5,
    help="Number of random normals to generate [default %(default)s]",
    metavar="number")
parser$add_argument("--generator", default="rnorm",
    help = "Function to generate random deviates [default \"%(default)s\"]")
parser$add_argument("--mean", default=0, type="double",
    help="Mean if generator == \"rnorm\" [default %(default)s]")
parser$add_argument("--sd", default=1, type="double",
        metavar="standard deviation",
    help="Standard deviation if generator == \"rnorm\" [default %(default)s]")

# get command line options, if help option encountered print help and exit,
# otherwise if options not found on command line then set defaults,
args <- parser$parse_args()

# print some progress messages to stderr if "quietly" wasn't requested
if ( args$verbose ) {
    write("writing some verbose output to standard error...\n", stderr())
}

# do some operations based on user input
if( args$generator == "rnorm") {
    cat(paste(rnorm(args$count, mean=args$mean, sd=args$sd), collapse="\n"))
} else {
    cat(paste(do.call(args$generator, list(args$count)), collapse="\n"))
}
cat("\n")
```

By default *argparse* will generate a help message if it encounters `--help` or `-h` on the command line. Note how `%(default)s` in the example program was replaced by the actual default values in the help statement that *argparse* generated.

```
bash$ example.R --help
```

```
usage: example.R [-h] [-v] [-q] [-c number] [--generator GENERATOR]
                 [--mean MEAN] [--sd standard deviation]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Print extra output [default]
  -q, --quietly         Print little output
  -c number, --count number
                        Number of random normals to generate [default 5]
  --generator GENERATOR
                        Function to generate random deviates [default "rnorm"]
  --mean MEAN           Mean if generator == "rnorm" [default 0]
  --sd standard deviation
                        Standard deviation if generator == "rnorm" [default 1]
```

If you specify default values when creating your `ArgumentParser` then *argparse* will use them as expected.

```
bash$ example.R
```

```
writing some verbose output to standard error...

-0.29350292053147
-0.275584012963682
-1.67512837378289
-0.512607468866483
-0.399912145435721
```

Or you can specify your own values.

```
bash$ example.R --mean=10 --sd=10 --count=3
```

```
writing some verbose output to standard error...

4.09288833878112
11.5601386630138
-3.58133473029068
```

If you remember from the example program that `--quiet` had `action="store_false"` and `dest="verbose"`. This means that `--quiet` is a switch that turns the `verbose` option from its default value of `TRUE` to `FALSE`. Note how the `verbose` and `quiet` options store their value in the exact same variable.

```
bash$ example.R --quiet -c 4 --generator="runif"

0.834299867972732
0.137762663885951
0.265378064941615
0.351195215014741
```

If you specify an illegal flag then *argparse* will print out a usage message and an error message and quit.

```
bash$ example.R --silent -m 5

usage: example.R [-h] [-v] [-q] [-c number] [--generator GENERATOR]
                 [--mean MEAN] [--sd standard deviation]
example.R: error: unrecognized arguments: --silent -m 5
```

If you specify the same option multiple times then *argparse* will use the value of the last option specified.

```
bash$ example.R -c 100 -c 2 -c 1000 -c 7

writing some verbose output to standard error...

0.0664575640768572
-1.04258166614327
-0.333155469453426
1.56752856858253
0.804976315382004
-1.75469258950457
0.0628950128221838
```

argparse can also parse positional arguments. Below we give an example program *display_file.R*, which is a program that prints out the contents of a single file (the required positional argument, not an optional argument) and which accepts the normal help option as well as an option to add line numbers to the output.

```
bash$ display_file.R --help

usage: display_file.R [-h] [-n] file

positional arguments:
  file               File to be displayed

optional arguments:
  -h, --help         show this help message and exit
  -n, --add_numbers  Print line number at the beginning of each line [default]

bash$ display_file.R --add_numbers display_file.R

1 #!/usr/bin/env Rscript
2 suppressPackageStartupMessages(library("argparse"))
3
4 parser <- ArgumentParser()
5 parser$add_argument("-n", "--add_numbers", action="store_true", default=FALSE,
6     help="Print line number at the beginning of each line [default]")
7 parser$add_argument("file", nargs=1, help="File to be displayed")
8
9 args <- parser$parse_args()
10
11 file <- args$file
12 # if(length(arguments$args) != 1) {
13 #     cat("Incorrect number of required positional arguments\n\n")
14 #     print_help(parser)
15 #     stop()
16 # } else {
17 #     file <- arguments$args
18 # }
19
20 if( file.access(file) == -1) {
21     stop(sprintf("Specified file ( %s ) does not exist", file))
22 } else {
23     file_text <- readLines(file)
24 }
25
26 if(args$add_numbers) {
27     cat(paste(1:length(file_text), file_text), sep = "\n")
28 } else {
29     cat(file_text, sep = "\n")
30 }

bash$ display_file.R non_existent_file.txt

Error: Specified file ( non_existent_file.txt ) does not exist
Execution halted
```

```
bash$ display_file.R

usage: display_file.R [-h] [-n] file
display_file.R: error: too few arguments
```