optparse Command Line Option Parsing

August 4, 2010

optparse is a command line option parser inspired by Python's "optparse" library. Use this with Rscript to write "#!"-shebang scripts that accept short and long flags/options, 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.Rscript" and "display_file.Rscript" illustrating the use of the optparse package.

```
bash$ ls
```

```
display_file.Rscript
example.Rscript
optparse.Rnw
optparse.tex
```

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.Rscript example.Rscript
bash$ export PATH=$PATH:`pwd`
```

Here is what “example.Rscript” contains:

```
bash$ display_file.Rscript example.Rscript

#!/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("optparse"))

# specify our desired options in a list
# by default OptionParser will add an help option equivalent to
# make_option(c("-h", "--help"), action="store_true", default=FALSE,
#               help="Show this help message and exit")
option_list <- list(
    make_option(c("-v", "--verbose"), action="store_true", default=TRUE,
        help="Print extra output [default]"),
    make_option(c("-q", "--quietly"), action="store_false",
        dest="verbose", help="Print little output"),
    make_option(c("-c", "--count"), type="integer", default=5,
        help="Number of random normals to generate [default %default]",
        metavar="number"),
    make_option("--generator", default="rnorm",
        help = "Function to generate random deviates [default \"%default\"]"),
    make_option("--mean", default=0,
        help="Mean if generator == \"rnorm\" [default %default]"),
    make_option("--sd", default=1, metavar="standard deviation",
        help="Standard deviation if generator == \"rnorm\" [default %default]")
    )

# get command line options, if help option encountered print help and exit,
# otherwise if options not found on command line then set defaults,
opt <- parse_args(OptionParser(option_list=option_list))

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

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

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

```
bash$ example.Rscript --help

usage: example.Rscript [options]

options:
        -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]

        -h, --help
                Show this help message and exit
```

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

```
bash$ example.Rscript

writing some verbose output to standard error...

-0.478589396597093
0.203533366540883
-0.049683668562277
-0.361529229925173
0.17828581151579
```

Or you can specify your own values.

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

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

-8.16075400179687
6.71537695784529
28.3485993858317
```

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.Rscript --quiet -c 4 --generator="runif"
```

```
0.634035702329129
0.284539718879387
0.492543415399268
0.939225726062432
```

If you specify an illegal flag then *optparse* will throw an error.

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

```
Error in getopt(spec = spec, opt = args) : long flag "silent" is invalid
Calls: parse_args -> getopt
Execution halted
```

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

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

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

1.45668567265132
0.00343826316541492
0.761889243763321
-0.0341608126882785
0.125666980957272
-1.28494255053503
-1.36127511860933
```

optparse can also recognize positional arguments if `parse_args` is given the option `positional_arguments = TRUE`. Below we give an example program *display_file.Rscript*, 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. Note that the positional arguments need to be placed *after* the optional arguments.

```
bash$ display_file.Rscript --help

display_file.Rscript [options] file

options:
        -n, --add_numbers
                Print line number at the beginning of each line [default]

        -h, --help
                Show this help message and exit

bash$ display_file.Rscript --add_numbers display_file.Rscript

1 #!/usr/bin/env Rscript
2 suppressPackageStartupMessages(library("optparse"))
3
4 option_list <- list(
5     make_option(c("-n", "--add_numbers"), action="store_true", default=FALSE,
6         help="Print line number at the beginning of each line [default]")
7     )
8 parser <- OptionParser(usage = "%prog [options] file", option_list=option_list)
9
10 arguments <- parse_args(parser, positional_arguments = TRUE)
11 opt <- arguments$options
12
13 file_text <- readLines(arguments$args)
14 if(opt$add_numbers) {
15     cat(paste(1:length(file_text), file_text), sep = "\n")
16 } else {
17     cat(file_text, sep = "\n")
18 }
```