optparse Command Line Option Parsing

October 28, 2009

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.

Here is an example R script illustrating the use of the optparse package.

```
#!/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")
```

To avoid having to worry about correctly specifying our `"#!"`-shebang line and making our program executable we will change to the directory of our example program and use `Rscript` directly.

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$ Rscript 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$ Rscript example.Rscript

writing some verbose output to standard error...

0.79343782274519
-0.436112097397407
0.269025918368965
-0.68747434814307
-1.59915200433100
```

Or you can specify your own values.

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

writing some verbose output to standard error...

19.8547727542335
20.0449305059524
15.5352647430356
```

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$ Rscript example.Rscript --quiet -c 4 --generator="runif"

0.74183233268559
0.533134274883196
0.907446366269141
0.418517381884158
```

If you specify an illegal flag then *getopt*, the package *optparse* uses to do the actual command line parsing, will throw an error. *getopt* will also throw an error if you specify two options that use identical flags.

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

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