

optparse Command Line Option Parsing

October 19, 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...

-2.81414802387397
-0.847688370604817
-0.440243471749738
-1.09053285803243
-1.51213745991914
```

Or you can specify your own values.

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

writing some verbose output to standard error...

```
3.45482649547608
11.8930150115807
14.9373291076918
```

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"
```

```
0.291339703835547
0.718197723617777
0.567863655276597
0.135132995434105
```

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
```