optparse Command Line Option Parsing

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.R" and "display_file.R" illustrating the use of the optparse 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
# Copyright 2010-2013 Trevor L Davis <trevor.l.davis@stanford.edu>
# Copyright 2008 Allen Day
#
#  This file is free software: you may copy, redistribute and/or modify it
#  under the terms of the GNU General Public License as published by the
#  Free Software Foundation, either version 2 of the License, or (at your
#  option) any later version.
#
#  This file is distributed in the hope that it will be useful, but
#  WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
#  General Public License for more details.
#
#  You should have received a copy of the GNU General Public License
#  along with this program.  If not, see <http://www.gnu.org/licenses/>.
suppressPackageStartupMessages(library("optparse"))
suppressPackageStartupMessages(library("stats"))

# 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.R --help

```
Usage: example.R [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.R

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

0.816120523380514
-0.844405693064675
0.345035705986038
1.11855685384653
0.973655897322031
```

Or you can specify your own values.

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

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

13.6070232726135
25.8258533364187
-0.807588940171346
```

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"

```
0.580427136039361
0.791549972957
0.267960017547011
0.248713592300192
```

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

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

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

1.48483379693858
-0.470949597470633
1.80103728214831
1.416392178041
0.0159032762001856
0.426671496337673
-0.450357048478622
```

optparse can also recognize positional arguments if `parse_args` is given the option `positional_arguments = c(min_pa, max_pa)` where `min_pa` is the minimum and `max_pa` is the maximum number of supported positional arguments. (A single numeric corresponds to

`min_pa == max_pa`, `TRUE` is equivalent to `c(0, Inf)`, and `FALSE`, the default, is equivalent to `0`.) 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. Note that the positional arguments need to be placed *after* the optional arguments.

bash$ display_file.R --help

```
Usage: display_file.R [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.R --add_numbers display_file.R

```
1 #!/usr/bin/env Rscript
2 # Copyright 2010-2013 Trevor L Davis <trevor.l.davis@stanford.edu>
3 # Copyright 2013 Kirill Müller
4 #
5 #  This file is free software: you may copy, redistribute and/or modify it
6 #  under the terms of the GNU General Public License as published by the
7 #  Free Software Foundation, either version 2 of the License, or (at your
8 #  option) any later version.
9 #
10 #  This file is distributed in the hope that it will be useful, but
11 #  WITHOUT ANY WARRANTY; without even the implied warranty of
12 #  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
13 #  General Public License for more details.
14 #
15 #  You should have received a copy of the GNU General Public License
16 #  along with this program.  If not, see <http://www.gnu.org/licenses/>.
17 suppressPackageStartupMessages(library("optparse"))
18
19 option_list <- list(
20     make_option(c("-n", "--add_numbers"), action="store_true", default=FALSE,
21         help="Print line number at the beginning of each line [default]")
22     )
23 parser <- OptionParser(usage = "%prog [options] file", option_list=option_list)
24
25 arguments <- parse_args(parser, positional_arguments = 1)
26 opt <- arguments$options
27 file <- arguments$args
28
29 if( file.access(file) == -1) {
30     stop(sprintf("Specified file ( %s ) does not exist", file))
31 } else {
32     file_text <- readLines(file)
33 }
34
35 if(opt$add_numbers) {
36     cat(paste(1:length(file_text), file_text), sep = "\n")
```

```
37 } else {
38     cat(file_text, sep = "\n")
39 }
```

bash$ display_file.R non_existent_file.txt

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

bash$ display_file.R

```
Error in parse_args(parser, positional_arguments = 1) :
  required at least 1 positional arguments, got 0
Execution halted
```