Genome Analysis Toolkit (GATK) User Manual

Matt Hanna

30 Mar 2009

1 Walkers

1.1 Command-line Arguments

Users can create command-line arguments for walkers by creating public member variables annotated with @Argument in the walker. The @Argument annotation takes a number of different parameters:

fullName The full name of this argument. Defaults to the toLowerCase()'d member name. When specifying fullName on the command line, prefix a double dash (–).

shortName The alternate, short name for this argument. Defaults to the first letter of the member name. When specifying shortName on the command line, prefix a single dash (-).

doc Documentation for this argument. Will appear in help output when a user either requests help with the –help (-h) argument or when a user specifies an invalid set of arguments.

required Whether the argument is required when used with this walker. Default is required = true.

exclusive Specifies that this argument is mutually exclusive of another argument in the same walker. Defaults to not mutually exclusive of any other arguments.

defaultValue Specifies the default value for this parameter, in string form.

1.1.1 Passing Command-line Arguments

Arguments can be passed to the walker using either the full name or the short name. If passing arguments using the full name, the syntax is −−<arg full name> <value>.

```
--myint 6
```

If passing arguments using the short name, the syntax is -<arg short name><value>. Note that there is no space between the short name and the value:

```
-m6
```

Boolean (class) and boolean (primitive) arguments are a special in that they require no argument. The presence of a boolean indicates true, and its absence indicates false. The following example sets a flag to true.

```
-B
```

1.1.2 Examples

Create an required int parameter with full name –myint, short name -m. Pass this argument by adding "–myint 6" or -m6 to the command line.

```
import org.broadinstitute.sting.utils.cmdLine.Argument;

public class HelloWalker extends ReadWalker<Integer,Long> {
    @Argument
    public int myInt;
```

Create an optional float parameter with full name –myFloatingPointArgument, short name -m. Pass this argument by adding –myFloatingPointArgument 2.71 or -m2.71.

```
import org.broadinstitute.sting.utils.cmdLine.Argument;

public class HelloWalker extends ReadWalker<Integer,Long> {
    @Argument(fullName="myFloatingPointArgument",required=false,defaultValue="3.14159")
    public float myFloat;
```

The GATK will parse the argument differently depending on the type of the public member variable's type. Many different argument types are supported, including primitives and their wrappers, arrays, typed and untyped collections, and any type with a String constructor.

When the GATK cannot completely infer the type (such as in the case of untyped collections), it will assume that the argument is a String. GATK is aware of concrete implementations of some interfaces and abstract classes. If the argument's member variable is of type List or Set, the GATK will fill the member variable with a concrete ArrayList or TreeSet, respectively. Maps are not currently supported.

1.2 Output

By default, the walkers provide protected out and err PrintStreams. Users can write to these streams just as they write to System.out and System.err. This output can be redirected to a file using the out, err, and outerr command-line arguments.

1.3 Logging

The walkers provide a protected logger instance. Users can adjust the debug level of the walkers using the -l command line option.

Turning on verbose logging can produce more output than the user really needs. To selectively turn on logging for a class or package, the user can specify a log4j.properties property file from the command line as follows:

```
-Dlog4j.configuration=file:///humgen/gsa-scr1/hanna/src/Sting/java/config/log4j.properties
```

An example log4j.properties file is available in the java/config directory of the subversion repository.