SubOpt.pm – Odyssey Subroutine Input API

of G-language Genome Analysis Environment Version 1

1. AUTHOR

Kazuharu Arakawa, gaou@g-language.org

2. LAST UPDATE OF THIS DOCUMENT

March 21, 2002

3. INTRODUCTION

SubOpt.pm provides a basic API for Odyssey Subroutine Input. Because the analysis methods provided in G-language GAE that is known as Odyssey should be generic and simple to develop, this API provides an easy and unique access to the subroutine. This API is used in a very similar manner as the GetOpt module packaged with Perl. The difference is that this API is for subroutine options and not the program options, and that this API has special support for genome data. This API does not only provide ease for development, but also passes progress messages to the system core; therefore it is essential to create the Odyssey Subroutines using this API.

4. SYNOPSIS

```
use SubOpt;

&odyssey ($gb, -option=>"neat!", -file=>"fancy.txt", "oops");

sub odyssey{
    opt_default(option => 'none', file => 'hoge.txt');
    # Set default values. This is optional.

    my @args = opt_get(@_);
    # Parse options.

    my $gb = opt_as_gb(shift @args);
    # Get the input value as a G instance
```

```
    my $last = shift @args;

    print $gb->{SEQ} , "¥n";
    print "last: $last¥n";
    print "option: ", opt_val("option"), "¥n";
    print "file: ", opt_val("file"), "¥n";
    # Option values are accessed via opt_val().
}
```

5. DESCRIPTION

SubOpt parses the arguments given to a subroutine in a similar manner as GetOpt module.

Options are specified in the form: -option=>"hoge"
The above will input the value "hoge" with a key "option".
i.e. option with '-' is stored with the value pointed with '=>'.

Supported methods:

opt_default(<option>=><value>)

This mehod sets default values for the options.
i.e. if the option is not specified, the value set with this method is used. This method is optional.

The option name here should be enclosed in quotations, and do not have to have a hyphen in the beginning.

eg. opt_default("option"=>"hoge");

```
    #inputs default value of "hoge" for key "option"
    #if the subroutine is called with -option, like
    #&subroutine($gb, -option=>"boo");
    #the option value will be overridden by "boo"
```

opt_get(@_)

This method parses the option arguments, stores the options in its class, and returns the array of arguments that are not options.

eg. my @args = opt_get(@_);

#receives all arguments in @args, and all options are stored

#in SubOpt name space; therefore options are not passed to @args

opt_val(<option>)

This method returns the value of the given option. The option name here should be enclosed in quotations.

eg. my $key = opt_val("key");

#gets the value for the option "key".

#It is recommended to store the option in a local variable

#instead of calling opt_val() everytime the option is used.

#This is for the prevention of the confusion of SubOpt name space

#in case SubOpt is called in cascade.

opt_as_gb(<$gb or $seq>)

This method automatically distinguish the variable from a scalar sequence, reference of a sequence, and a G instance, and returns the value as a G instance. If the input variable is a scalar of reference of sequence, the G instance only contains the $gb->{SEQ} object.

If only the sequence portion of the G instance is required in a subroutine, &subroutine($gb); and &subroutine($gb->{SEQ}); and &subroutine(¥$gb->{SEQ}); and &subroutine("atgc"); and should all be used by the subroutine. In this case, calling opt_as_gb() as

my @args = opt_get(@_);

my $gb = opt_as_gb(shift @args);

will store the first argument in G instance form regardless of the type of input argument. Therefore, the sequence information is accessible as $gb->{SEQ} in the subroutine.