NEAT

Adding modules

Patrick Schorderet
Patrick.schorderet@molbio.mgh.harvard.edu
June 2015
1 INTRODUCTION ................................................................................................................................................. 4

2 ARCHITECTURE .................................................................................................................................................. 5
  2.1 GENERAL ARCHITECTURE ............................................................................................................................ 5
  2.2 CODE ARCHITECTURE .................................................................................................................................... 6
1 Introduction

NExt generation Analysis Toolbox (NEAT) is a perl/R package that supports users during the analysis of next generation sequencing (NGS).

NEAT is versatile and easy to modify. In this tutorial, we will show how to add a custom module to NEAT. Adding a new module has been made as easy as possible by automating all the repetitive tasks such as job creation, batch submission and queuing. Adding a new module usually falls down to a single line of code.
2 Architecture

2.1 General architecture

NEAT contains different modules (yellow boxes) than can be modified and/or added.

Fig.1 NEAT architecture. NGS data can be analyzed using NEAT in less than a day. Users follow a logical 4-step process, including the creation of a new project, running the pipeline on a remote server or in the cloud, transferring the data to a local computer and proceeding to the analysis.
2.2 Code architecture

The main code of NEAT (in the example of ChIPseq projects) is found in 
./NEAT/ChIPpip/scripts/ChIPpip.pl. The code is well annotated, highly redundant and shuld be self-explanatory to advanced users. Main modules are easily identifiable and customizable. A brief summary of how each module is built is depicted below.
Map module

If job is TRUE, enter module
(modify the 'Steps_to_execute variables at the top of the code)

Name your job (Tex map)

A file is created with YOUR_JOB_NAME. This file will contain the flow of each individual job related to this job.

For loop that iterates through all samples and inputs. To iterate through samples only, use the @samples table instead of the @samplesInputs table.

For each individual job, create a qsub-like file (in which the job code will be) by copying the qsub header ($srchead) into the intermediary $SUBInt file.

YOUR CUSTOMIZE CODE GOES HERE

Keep track of job names to ensure job dependencies (see below)

Modify the Targets.txt file to mark the job as '.DONE'

Prepare code for the next iteration ($iterateSH). This iteration depends on the completion of all the current jobs ($myJobsVec)

Submit job to cluster and exit
To add a module, copy-paste an entire block and change the following:

- Name of job ($jobName)
- Make sure the loop is correctly set (looping over all samples, inputs, etc)
- Add your code line under the “important code here” section
- Lastly, add the variable to the others in the “# Steps_to_execute_pipe” section at the top portion of the code.