# NGS-FABLAB: Designing Webpages and Integrating QIIME

## Internship Report - I

11/12/2014

*Submitted by*

**Ashok Varadharajan**Msc Epidemiology

IBE – Institute for medical Informatics, Biometry and Epidemiology

Ludwig-Maximilians-University (LMU) Munich

*Supervised by*

Sebastian Schaaf

IBE – Institute for medical Informatics, Biometry and Epidemiology

Ludwig-Maximilians-University (LMU) Munich

# Table of Contents

## 1. INTRODUCTION

High-throughput sequencing, which is also termed as Next generation Sequencing (NGS), enabled the researchers to sequence the large genomic data at very low cost. Analysis of such large genomic data requires computationally intensive, flexible and user friendly platforms. Galaxy(1) is one of the web-based frameworks for analysis of NGS data and has been implemented at IBE, LMU as "NGS-Fablab", supporting intensive biomedical research. NGS-Fablab supports various projects related to Genetic Epidemiology and Bioinformatics, which are my areas of interest. The current report summarizes the tasks which were allotted to me during my internship of about nine weeks from 1st April, 2014 to 31st July, 2014.

## 2. TASKS

Following tasks were assigned to me during this internship.

1. Designing user-friendly, policy-conform webpages for NGS-Fablab. It acts as a welcome page, gives an overview and further information about the local Galaxy infrastructure, affiliated institutions, local guidelines, an attached data registry, a wiki board and related publications.
2. Integrating the QIIME (Quantitative Insights into Microbial Ecology) tool box into NGS-Fablab.

## 3. METHODS

### 3.1 Task 1: Designing webpages for NGS-Fablab

Various technologies are available to design webpages. Since the current project involves only static webpages, tools such as HTML, CSS and JavaScript are enough to construct the required webpages. Hypertext Markup Language (HTML) is nothing but standard markup language to create webpages. Browsers like Firefox and Internet Explorer interpret such markup language and display the required contents to the user. Cascading Style Sheets (CSS) is a style sheet language that helps to customize the look of the webpage written in a markup language. JavaScript is a computer programming language that supports interactive effects within browsers. JQuery is a collection of JavaScript libraries. It can make much easier to implement JavaScript in html. jQuery-UI and jQuery bxSlider are those libraries used in the current webpages designing. Corporate Design Manual of both LMU University (2) and LMU Klinikum (3) have been used as a reference for creating pages, which meet the guidelines. GNU Image Manipulation Program (GIMP) has been used for creating the header images and environment images (such as labels for "Development", "Production" and "Testing" environment).

## 3.2 Task 2: Integrating QIIME tool in to Galaxy framework of NGS-Fablab.

### 3.2.1 QIIME

QIIME stands for "Quantitative Insights Into Microbial Ecology". QIIME is an open source software package for analysis of microbial sequence data generated on a variety of platforms. Typical data sets are SSU rRNA and shotgun metagenomic data. QIIME is nothing but a collection of python scripts, which interconnect common command line tools such as tax2tree, blast, cd-hit, chimeraSlayer, mothur, usearch etc. in order to generate small workflows. Hence, every "tool" from this toolbox is already a pipeline in itself. These pipelines are dedicated to support various analyses like OTU picking, taxonomic assignment and phylogenetic trees construction. QIIME has been applied to studies based on billions of sequences from thousands of samples. (4-8)

There are several ways to get a working install of QIIME.

1. Using Virtual Machines, which are preloaded with QIIME and its dependencies
2. Automated installation of QIIME using **qiime-deploy** tool
3. Native installation of QIIME, where all the required tools are installed manually

In NGS-Fablab, the qiime-deploy tool has been used to install QIIME software and all its dependencies.

### 3.2.2 Qiime-deploy tool

Qiime-deploy is a Python-based automated tool, for building, configuration, and deployment of QIIME. It can be downloaded from the GIT repository by phusemen (9). Python scripts in Qiime tool cannot run on its own and a lot of dependant tools are required to be installed in the system. Hence, a main task for qiime-deploy is resolving those dependencies on Linux systems automatically. The qiime-deploy tool depends on a configuration file, namely qiime-deploy-conf (10), which provides information of all dependant tools required by qiime such as download links, build types, configure parameters, install options etc. Both qiime and qiime-deploy are downloaded from git by using the below code.

```
$ cd ${QIIME_INSTALL_DIR}

$ git clone https://github.com/phuseman/qiime-deploy.git

$ cd ${QIIME_INSTALL_DIR}/qiime-deploy

$ git pull origin patch-1

$ git clone https://github.com/qiime/qiime-deploy-conf.git
```

### 3.2.3 Qiime-galaxy tool

Qiime-galaxy tool is a python application developed by josenavas (11) in order to automatically integrate QIIME on Galaxy. The QIIME-galaxy tool can be downloaded from the following GIT repository. Before running qiime-galaxy tool, QIIME and Galaxy should be installed successfully in the system. The download from GIT can be achieved by using the following code.

```
git clone https://github.com/qiime/qiime-galaxy.git
```

### 3.2.4 Steps for integrating QIIME and Galaxy in NGS-Fablab

Though there are automated tools available for integrating QIIME and Galaxy, in practice it requires lot of modifications, configurations, customization and additional packages to be done to make a successful integration. The process has been divided in several steps mentioned below for better understanding.

1. Installation of dependant internal packages required for both qiime-deploy and qiime-galaxy tool.
2. Modification of qiime-deploy-conf required for qiime-deploy tool.
3. Installation of QIIME using qiime-deploy tool.
4. Integration of QIIME and Galaxy using qiime-galaxy tool.

### 3.2.5 Automated shell script for QIIME-Galaxy integration

In each and every step mentioned above, there are several lines of Linux commands to be executed. The local development policy for NGS-FabLab is to provide fully automated code snippet in order to enable developers to include a new piece of software each and every time a (development) instance is created. Therefore, automated shell scripts have been developed to do those steps. An overview of automated scripts can be seen in Figure 1. Description and usage of these automated scripts are described below. Entire code of these automated scripts could be found in the result pages.


**Figure 1: Overview of the automated shell scripts**

***## install_packages_qiime.sh - install the required packages mentioned in the step1 above.***

***## integrate_qiime.sh - install qiime using and qiime-deploy tools and integrate it using qiime-galaxy tool (step3 and step4)***

***## QIIME-Galaxy_integration_main.sh - just calling both install_packages.sh and integrate_qiime.sh***

***## how to run the script?***

***$ sh QIIME-Galaxy_integration_main.sh***

### 3.2.6 Enabling Workflow for Qiime in Galaxy

After successful integration of Qiime and Galaxy, qiime scripts can be found in the tool menu of the web-based user interface of Galaxy. The user can just select the tools by a click and perform the analysis of an input data set within the Galaxy framework. One important feature of Galaxy is the construction of complex workflows, where several tools are interconnected sequentially and act as a complex modular pipeline. The integrations done by the qiime-galaxy tools do not facilitate the qiime tools to be connected to the other tools in the galaxy workflow. This is due to the fact that the default output of the several qiime scripts is a directory of files, which is compressed to a single ".tgz" file. This cannot be used as an input to subsequent Galaxy modules anymore.

To solve this problem, an additional wrapper, namely "qiime_connector", has been developed which will act a bridge between QIIME outputs in Galaxy and subsequent modules. This enables the user to use qiime in galaxy workflows. The entire script of the qiime_connector can be seen below. Comparison of qiime workflow before and after modification is illustrated in Figure 2.

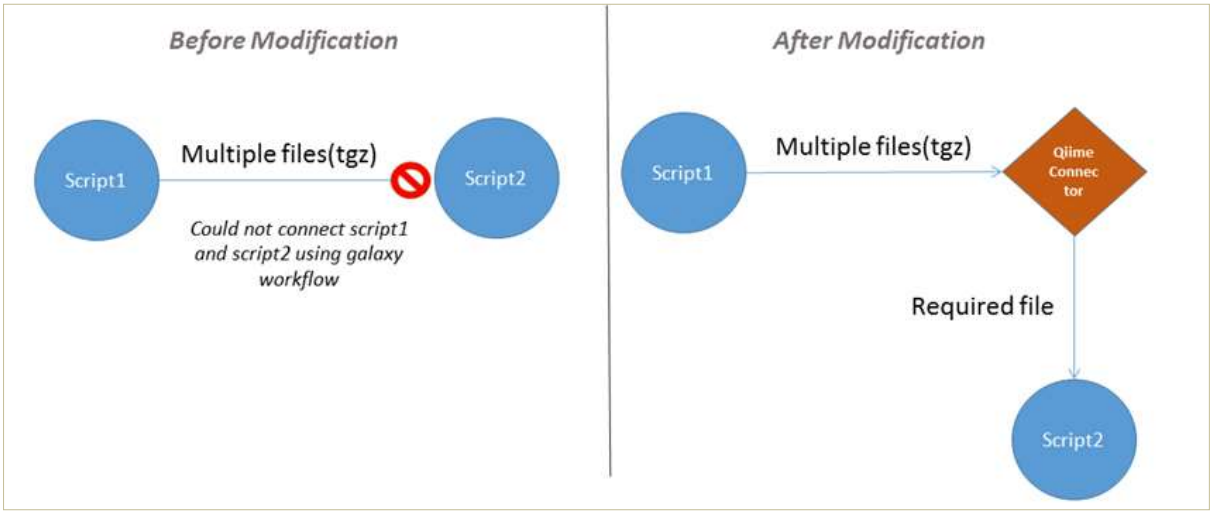

**Figure 2: Comparison of the Qiime workflow in Galaxy before and after Modification**

```
## QIIME_CONNECTOR.XML

<tool id="qiime_connector" name="QIIME connector" version="1.0.0">
  <description></description>
  <command>
    source /home/galaxy/galaxy-dist/tools/qiime_connector/qiime_connector.sh
${input} ${input.id} ${__new_file_path__} ${output1} ${file_name_1}
  </command>
  <inputs>
    <param name="input" type="data" format="tgz" label="qiime output file"/>
       <param name="file_name_1" type="text" value="" label="File1: enter file
name with extension" />
       <param name="out_format_1" type="select" label="Output data type">
                 <option value="txt">txt</option>
                 <option value="fna">fna</option>
                 <option value="biom">biom</option>
                 <option value="tre">tre</option>
       </param>
  </inputs>
  <outputs>
    <data format="txt" name="output1">
      <change_format>
        <when input="out_format_1" value="fna" format="fna" />
        <when input="out_format_1" value="biom" format="biom" />
        <when input="out_format_1" value="tre" format="tre" />
      </change_format>
    </data>
  </outputs>
</tool>

## QIIME_CONNECTOR.SH
input=$1
inputid=$2
nfp=$3
output1=$4
infile1=$5
inputfilename=`basename ${input} .dat`
echo ${inputfilename}
echo ${inputid}
echo ${output1}
echo ${infile1}
echo ${nfp}
temp=${nfp}/${inputfilename}'_'${inputid}
mkdir -p ${temp}
echo ${temp}
tar xvf ${input} -C ${temp} --touch
i=1
for filename in ${infile1};
       do
                 echo ${i}
                 OUTPUT_PATH=`eval echo '$'output$i`
                 echo ${OUTPUT_PATH}
                 echo ${filename}
                 if [ "${OUTPUT_PATH}" != "None" ]; then
                           echo `find ${temp} -name "$filename"`
                           for f in `find ${temp} -name "$filename"`;
                                     do
```

```
                                        cp ${f} ${OUTPUT_PATH}
                                    done
                fi
                ((i++))
        done
rm -R ${temp}
```

# 4. RESULTS

## 4.1 Task 1: Designing webpages for NGS-Fablab

Welcome page for NGS-Fablab has been designed. As mentioned above HTML, CSS and jQuery have been used to develop all the webpages. These pages will act as landing page for the user, before they do any kind of analysis in NGS-Fablab. Each webpage has six divisions such as Header, left content, main content, Right content and Footer. Header image was designed as per corporate design guidelines using GIMP image editor. Left content is the navigation area containing list of important titles such as Data Registry, Analysis Platform, Galaxy Main Resources, publications etc. related to NGS-Fablab. Main content displays all required information for each title. User can click any title in the navigation area (Left content), so that corresponding information will be displayed in the main content. Right Content has two sections - Environment section and Affiliation section. Environment section will prompt the working environment such as testing, development or production, to the user so that he/she would be cautious while working. Second section has a list of affiliated institutions which has collaboration with NGS-Fablab. If the User clicks any link from the list, they will be redirected to respective institution's official page. Footer contains certain common legal information like contact, imprint, terms and conditions.

### 4.1.1 Web Design – HOME PAGE

The user would be redirected when he/she click "HOME" link in the left content as shown in figure 3. This page presents a welcome message and a short overview about NGS-Fablab. Additionally, it contains image slider which was developed using jQuery plugin, namely bxSlider. Image slider facilitates slide animations for the give set of images. Later, admin can just replace those images with the newer ones and there is no change in html code required.

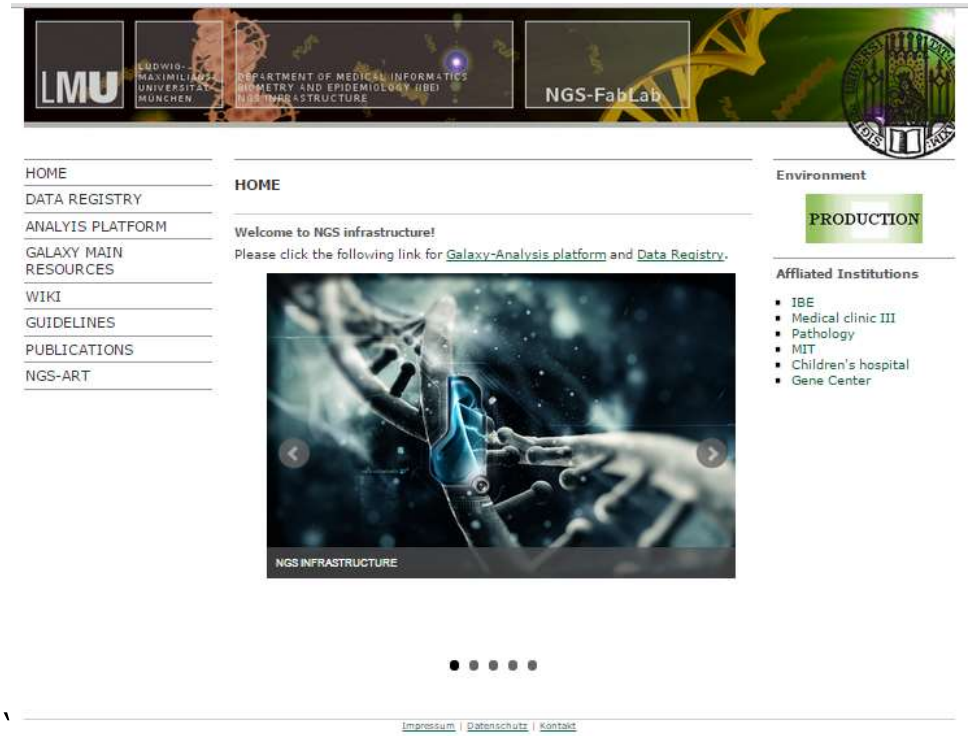


**Figure 3: HOME PAGE**

### 4.1.2 Web Design – DATA REGISTRY

The user can navigate to the galaxy platform from analysis platform which is shown in figure 5. This page has been created based on the idea that by clicking this link user would ask for login credentials. And the user who entered valid credentials should be allowed to use galaxy platform for their analysis.

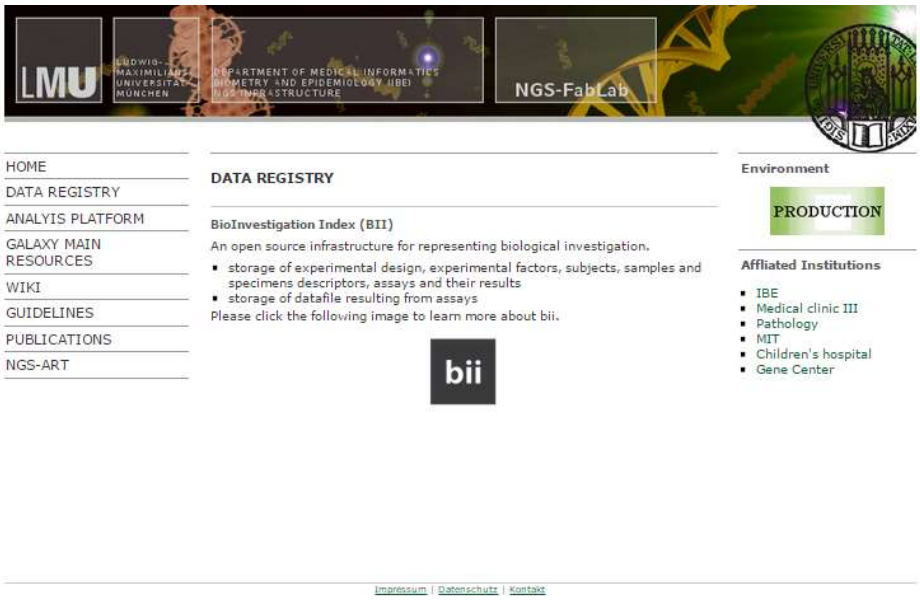


**Figure 3: DATA REGISTRY**

### 4.1.3 Web Design – GALAXY MAIN RESOURCES

Figure 5 shows the page which has information and useful materials from Galaxy main resources. These useful materials from the main galaxy could be provided in the form of links or attachments.

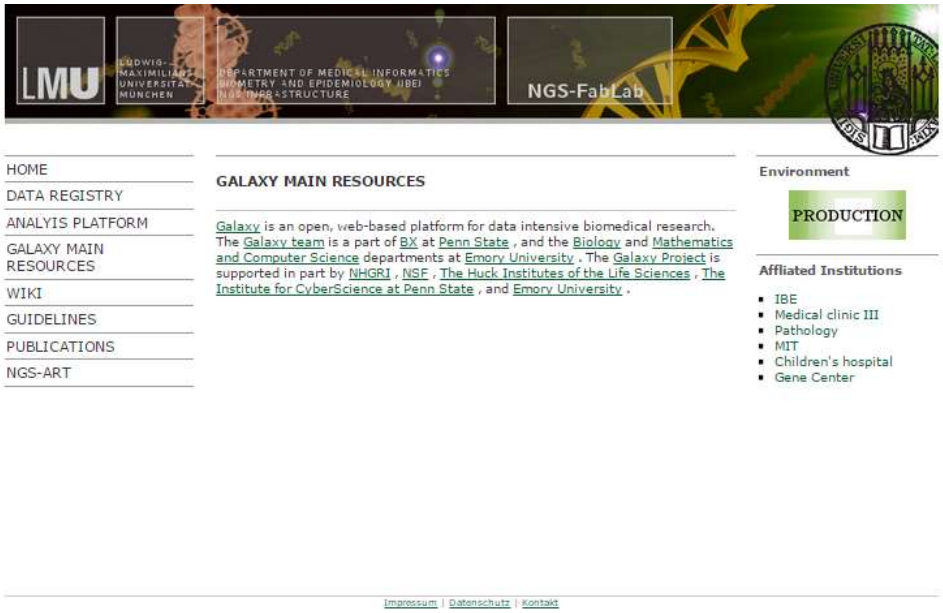


**Figure 5: GALAXY MAIN RESOURCES**

### 4.1.3 Web Design – WIKI

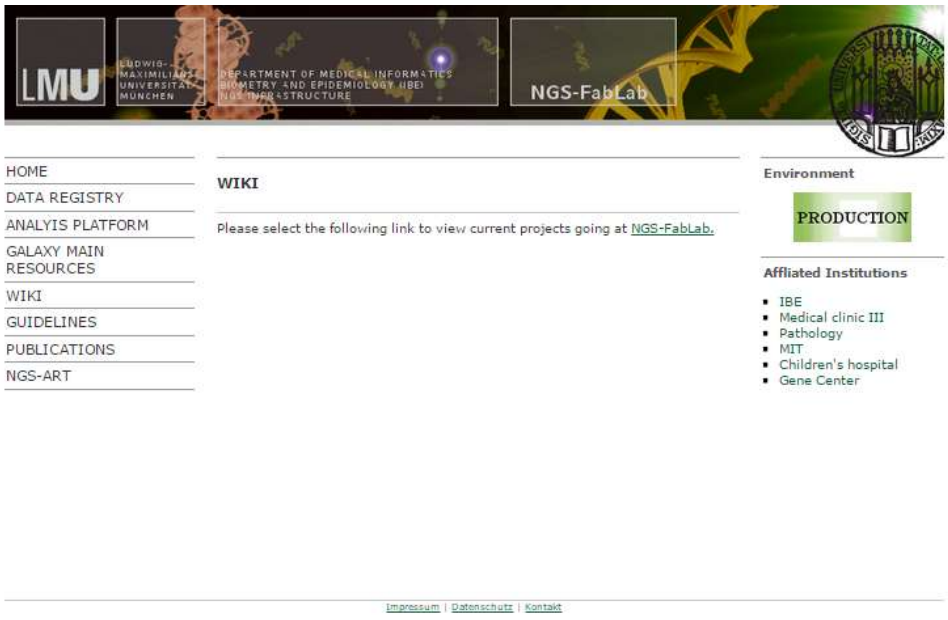


**Figure 7: WIKI**

Figure 7 shows wiki page for NGS-Fablab. This page will have all the training tutorials and guide on how to use NGS-Fablab galaxy platform. Admin can also add best practices tips for certain analysis if required. It would be very useful for the new user who does not know anything about galaxy platform or Next Generation Sequencing (NGS) analysis.

## 4.2 Task 2: Integrating QIIME tool in to Galaxy framework of NGS-Fablab

QIIME has been successfully integrated to galaxy using the tools such as Qiime-deploy, qiime-deploy-conf and qiime-galaxy. After successful integration, user can see qiime scripts in galaxy toolbox and do the required analysis similar to the one they did using command line operations. To make qiime integration very easy, automated script has been developed. Entire coding of these scripts can be found in supplementary section. Figure 3 shows snapshot of qiime scripts used in galaxy workflow. Qiime connector in the figure 3 will act as bridge between the two qiime scripts. Generally, it will pick the required file from the output (*.tgz) using the filename given by the user. Now the user can connect multiple scripts and utilize the functionality and advantage of galaxy workflow features.

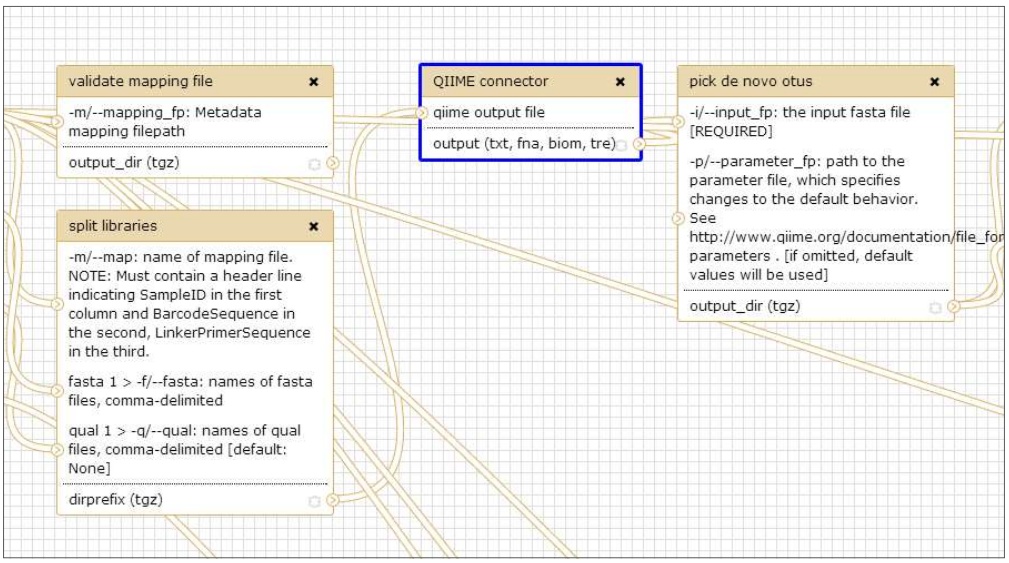


**Figure 7: WIKI**

## 5. DISCUSSION

Webpages shown in the results page are just templates. Content of these webpages are still need to be updated through proper discussion with concerned people. Previously, before these webpages were designed, all the users were redirected directly to the galaxy analysis platform without knowing much about NGS-Fablab. Now the users would get a brief knowledge about NGS-Fablab through these webpages. And also they can be benefited from the guidelines and tutorials available in the wiki page. Wiki page is still need to be updated and not it has only a link to Redmine project management system for NGS-Fablab. Environment section will display the current environment being used and will prompt the developers to be cautious if he/she is dealing with production environment. These webpages will also benefit admin for screening unauthorized users. But such feature is still need to be implemented.

In the other task, integration of QIIME tool in to galaxy, now users do not need to worry about executing complex commands in the command line. They can just do their analysis in a web based framework which is very simple and convenient. They can utilize all the galaxy features to analyze with qiime scripts such as re-running jobs, managing histories, workflow etc. Automated scripts were developed for the entire complex process of QIIME integration to galaxy. These scripts will enable admins for integration in several machines without any manual interference, the same holds for developers who setup a whole virtual system, reflecting the production machine as close as possible. Tools originally used for the integration do not facilitate the use of qiime scripts in galaxy workflow. To overcome this issue, Qiime_connector tool was created.

For me myself as an internship student, concepts of Galaxy, Galaxy workflows, HTML, JavaScript, CSS, GIMP, jQuery and UNIX commands (shell scripts) were learned through this task. While Galaxy is a widely distributed framework in bioinformatics, UNIX and web technologies are even more generous. Both may come in my future career. On top, the principle of development circles was conveyed. Finally, I learned lots of new things through the challenges and issues faced during this period.

# 6. SUPPLEMENTARY

## 6.1 QIIME-Galaxy_integration_main.sh

```
##Installing packages as root user
echo "Given mode:${1}"
sudo sh install_packages_qiime.sh

# Reloadin shared libraries
sudo /sbin/ldconfig

##Installing QIIME as  Galaxy user
sudo –u galaxy sh integrate_qiime.sh ${1}
```

## 6.2 install_packages_qiime.sh

```
function zypper_checkInstall {

   ZYPPER_TOOL_VAR=$1

   if [ ! "`zypper --gpg-auto-import-keys se --match-exact $ZYPPER_TOOL_VAR | grep -w
"package" | cut -f1 -d "|"`" = "i " ] ;
   then
      echo "Installing $ZYPPER_TOOL_VAR ..."
      zypper -n install $ZYPPER_TOOL_VAR
      echo "tool $ZYPPER_TOOL_VAR installed..."
   else
      echo "tool $ZYPPER_TOOL_VAR already exist..."
   fi

}
## Installing with force resolution option
function zypper_checkInstall_fc {

   ZYPPER_TOOL_VAR=$1

   if [ ! "`zypper --gpg-auto-import-keys se --match-exact $ZYPPER_TOOL_VAR | grep -w
                  "package" | cut -f1 -d "|"`" = "i " ] ;
   then
      echo "Installing $ZYPPER_TOOL_VAR ..."
      zypper -n install --force-resolution $ZYPPER_TOOL_VAR
      echo "tool $ZYPPER_TOOL_VAR installed..."
   else
      echo "tool $ZYPPER_TOOL_VAR already exist..."
   fi

}

function zypper_repo_checkInstall {

       ZYPPER_TOOL_REPO_LINK=$1
   ZYPPER_TOOL_VAR=$2

   if [ ! "`zypper --gpg-auto-import-keys se --match-exact $ZYPPER_TOOL_VAR | grep -w
                  "package" | cut -f1 -d "|"`" = "i " ] ;
   then
      echo "Installing $ZYPPER_TOOL_VAR ..."
      zypper -n --gpg-auto-import-keys -p $ZYPPER_TOOL_REPO_LINK -v install
                  $ZYPPER_TOOL_VAR
      echo "tool $ZYPPER_TOOL_VAR installed..."
   else
```

```
        echo "tool $ZYPPER_TOOL_VAR already exist..."
    fi
}

##installing with force resolution option
function zypper_repo_checkInstall_fc {

        ZYPPER_TOOL_REPO_LINK=$1
    ZYPPER_TOOL_VAR=$2

    if [ ! "`zypper --gpg-auto-import-keys se --match-exact $ZYPPER_TOOL_VAR | grep -w
                    "package" | cut -f1 -d "|"`" = "i " ] ;
    then
        echo "Installing $ZYPPER_TOOL_VAR ..."
        zypper -n --gpg-auto-import-keys -p $ZYPPER_TOOL_REPO_LINK -v install --force-
                    resolution $ZYPPER_TOOL_VAR
        echo "tool $ZYPPER_TOOL_VAR installed..."
    else
        echo "tool $ZYPPER_TOOL_VAR already exist..."
    fi
}



#Basic library installation
zypper -n in -t pattern sdk_c_c++
zypper -n in -t pattern Basis-Devel
zypper_checkInstall ant
zypper_checkInstall gcc
zypper_checkInstall gcc43-fortran
zypper_checkInstall gcc-fortran
zypper_checkInstall java-1_7_0-ibm-devel
zypper_checkInstall freetype
zypper_checkInstall freetype-tools
zypper_checkInstall freetype2
zypper_checkInstall freetype2-devel
zypper_checkInstall zlib
zypper_checkInstall zlib-devel
zypper_checkInstall mpich
zypper_checkInstall mpich-devel
zypper_checkInstall readline-devel
zypper_checkInstall gsl
zypper_checkInstall gsl-devel
zypper_checkInstall libxslt
zypper_checkInstall libpng-devel
zypper_checkInstall libpng12-0
zypper_checkInstall mysql
zypper_checkInstall mysql-tools
zypper_checkInstall libmysqlclient-devel
zypper_checkInstall xorg-x11-libXt
zypper_checkInstall xorg-x11-libXt-devel
zypper_checkInstall xorg-x11-libX11
zypper_checkInstall xorg-x11-libX11-devel
zypper_checkInstall libxml2
zypper_checkInstall xorg-x11-server
zypper_checkInstall dejavu
zypper_checkInstall python-devel
zypper_checkInstall sqlite3
zypper_checkInstall libsqlite3-0
zypper_checkInstall sqlite3-devel
zypper_checkInstall libncurses5
zypper_checkInstall ncurses-devel
zypper_checkInstall libbz2-devel
zypper_checkInstall libbz2-1
zypper_checkInstall git
zypper_checkInstall librdmacm
zypper_checkInstall liblapack3
zypper_checkInstall_fc blas
zypper_checkInstall_fc libblas3
zypper_checkInstall librdmacm-devel
zypper_checkInstall openmpi
```

```
zypper_checkInstall openmpi-devel
zypper_checkInstall xmlstarlet
zypper_repo_checkInstall
http://download.opensuse.org/repositories/home:/fengshuo:/zeromq/SLE_11_SP1/
zeromq
zypper_repo_checkInstall
http://download.opensuse.org/repositories/home:/fengshuo:/zeromq/SLE_11_SP1/
zeromq-devel
zypper_repo_checkInstall
http://download.opensuse.org/repositories/devel:/languages:/haskell/SLE_11_SP1/ ghc
zypper_repo_checkInstall
http://download.opensuse.org/repositories/home:/repabuild/SLE_11_SP1/ libatlas3-devel
zypper_repo_checkInstall
http://download.opensuse.org/repositories/home:/repabuild/SLE_11_SP1/ liblash1
zypper_repo_checkInstall
http://download.opensuse.org/repositories/home:/repabuild/SLE_11_SP1/ liblapack3
```

## 6.3 integrate_qiime.sh

```
#initializing variables

TEMP_DIR="/tmp/qiime_tmp"
QIIME_INSTALL_DIR="/home/galaxy/ext_tools/qiime"
QIIME_REPO_SCRIPTS_DIR="/home/HELIOS/avaradha/dev_home/qiime"
QIIME_REPO_TOOLS_DIR="/home/HELIOS/avaradha/dev_home/qiime/tools"
QIIME_SOFTWARE_DIR="${QIIME_INSTALL_DIR}/qiime_software"
GALAXY_DIR="/home/galaxy/galaxy-dist"
R_LIB_PATH="$(12)/tool_libs/R_libs"
QIIME_DEPLOY_LOG="${TEMP_DIR}/qiime-deploy.log"

LOCAL_PYTHON_LINK_PATH="/home/galaxy/ext_tools/python"
QIIME_WRAPPERS="${GALAXY_DIR}/tools/qiime1.8.0"
QIIME_SCRIPT_PATH="${QIIME_SOFTWARE_DIR}/qiime-1.8.0-release/bin"
QIIME_GALAXY_SCRIPT_PATH="${QIIME_INSTALL_DIR}/qiime-galaxy/scripts"

install_qiime(){

        echo "Inside install_qiime function"

        ## making directory for qiime installation
        mkdir -p "${QIIME_INSTALL_DIR}"
        cd ${QIIME_INSTALL_DIR}

        # Overriding existing pythonpath to none. (previous path disturbing the installation)
        echo -e '\n##Overriding existing pythonpath to none. (previous path disturbing the
                installation)' >> $HOME/.bashrc
        echo 'export PYTHONPATH=' >> $HOME/.bashrc
        echo -e '\n##Qiime' >> $HOME/.bashrc
        echo '#open-mpi path' >> $HOME/.bashrc
        echo 'export PATH=/usr/lib64/mpi/gcc/openmpi/bin:$PATH' >> $HOME/.bashrc
        echo 'export
LD_LIBRARY_PATH=/usr/lib64/mpi/gcc/openmpi/lib64:$LD_LIBRARY_PATH' >>
                $HOME/.bashrc
        source $HOME/.bashrc

        ## creating personal library for R

        mkdir -p ${R_LIB_PATH}
        echo '#R personal Library' >> $HOME/.bashrc
        echo 'export R_LIBS='${R_LIB_PATH}'' >> $HOME/.bashrc
        source $HOME/.bashrc



        ##### make sure qiime deploy config file is modified properly
        cp -R ${QIIME_REPO_SCRIPTS_DIR}/qiime-deploy-conf/ ${QIIME_INSTALL_DIR}/
```

```
        ##modify config file for qiime-deploy tool with path where all the tools are stored
        sed -i "s|<TOOLS_DIR>|${QIIME_REPO_TOOLS_DIR}|g"
                ${QIIME_INSTALL_DIR}/qiime-deploy- conf/qiime.conf

        ## qiime deploy
        cp -R ${QIIME_REPO_SCRIPTS_DIR}/qiime-deploy ${QIIME_INSTALL_DIR}/
        cd ${QIIME_INSTALL_DIR}/qiime-deploy
        python qiime-deploy.py ${QIIME_SOFTWARE_DIR}/ -f
                ${QIIME_INSTALL_DIR}/qiime-deploy-
                conf/qiime.conf --force-remove-failed-dirs >> ${QIIME_DEPLOY_LOG}
        source $HOME/.bashrc
        cat ${QIIME_DEPLOY_LOG}


}

create_python_links(){

        #creating python links
        echo "Inside create_python_links function"
        mkdir -p ${LOCAL_PYTHON_LINK_PATH}
        ln -s /usr/bin/python ${LOCAL_PYTHON_LINK_PATH}/python2.6.8
        ln -s ${LOCAL_PYTHON_LINK_PATH}/python2.6.8
                ${LOCAL_PYTHON_LINK_PATH}/python2.6
        ln -s ${QIIME_SOFTWARE_DIR}/python-2.7.3-release/bin/python
                ${LOCAL_PYTHON_LINK_PATH}/python2.7.3
        ln -s ${LOCAL_PYTHON_LINK_PATH}/python2.7.3
                ${LOCAL_PYTHON_LINK_PATH}/python2.7

        #default
        ln -s ${LOCAL_PYTHON_LINK_PATH}/python2.6
                ${LOCAL_PYTHON_LINK_PATH}/python
        echo '#Python Binary path' >> $HOME/.bashrc
        echo 'export PATH='${LOCAL_PYTHON_LINK_PATH}':$PATH' >> $HOME/.bashrc
        source $HOME/.bashrc



}

##Modify activate.sh file
modify_activate_sh(){

        echo "Inside modify_activate_sh function"
        sed -i 's/export LD_LIBRARY_PATH=/export
                LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/g'
                ${QIIME_SOFTWARE_DIR}/activate.sh
        source $HOME/.bashrc


}

##QIIME GALAXY INTEGRATION
integrate_on_galaxy(){

        echo "Inside integrate_on_galaxy function"
        cd ${QIIME_INSTALL_DIR}/
        #git clone https://github.com/qiime/qiime-galaxy.git
        cp -R ${QIIME_REPO_SCRIPTS_DIR}/qiime-galaxy/ ${QIIME_INSTALL_DIR}/
        cd ${QIIME_INSTALL_DIR}/qiime-galaxy
        echo '#qiime-galaxy path' >> $HOME/.bashrc
        echo 'export PATH='${QIIME_INSTALL_DIR}'/qiime-galaxy/scripts:$PATH' >>
                $HOME/.bashrc
        echo 'export PYTHONPATH='${QIIME_INSTALL_DIR}'/qiime-
                galaxy/lib:$PYTHONPATH' >> $HOME/.bashrc
        echo -e '\n' >> $HOME/.bashrc
        source $HOME/.bashrc
        ${LOCAL_PYTHON_LINK_PATH}/python2.7.3 ${QIIME_INSTALL_DIR}/qiime-
                galaxy/scripts/integrate_on_galaxy.py -i ${QIIME_SCRIPT_PATH} -g
                        ${GALAXY_DIR}/ -c  ${QIIME_INSTALL_DIR}/qiime-
                        galaxy/config_files/QIIME_1.8.0.conf --update_tool_conf
```

```
        ## reformating tool_conf.xml
        cp ${GALAXY_DIR}/tool_conf.xml ${TEMP_DIR}/
        xmllint --format ${TEMP_DIR}/tool_conf.xml > ${GALAXY_DIR}/tool_conf.xml

}

##QIIME CONNECTOR INTEGRATION
install_qiime_connector(){

        cp ${GALAXY_DIR}/tool_conf.xml ${TEMP_DIR}/
        x=`grep -c '<section.*id="qiime1.8.0".*' ${TEMP_DIR}/tool_conf.xml`
        y=`grep -c '<tool.*file="qiime_connector' ${TEMP_DIR}/tool_conf.xml`
        if [ ${x} -eq 0 ]
                then
                        echo "Qiime entry is not present in tool_conf.xml. Qiime connector
                                installation failed"
                        exit 1
                else
                        if [ ${y} -eq 0 ]
                        then
                                echo "Qiime entry is present in tool_conf.xml. Adding
                                        Qiime Connector entry"
                                sed '/<section id="qiime1.8.0"/a <tool
        file=\"qiime_connector/qiime_connector.xml\"/>'
                                        ${TEMP_DIR}/tool_conf.xml | xmllint –
                                                format - >
                                                ${GALAXY_DIR}/tool_conf.xml
                                cp -R ${QIIME_REPO_SCRIPTS_DIR}/qiime_connector/
                                        ${GALAXY_DIR}/tools/

                        else
                                echo "Qiime Connector entry is already present"
                        fi
                fi

}

##modify wrappers to use python 2.7.3
modify_wrappers(){

        echo "Inside modify_wrappers function"

        if [ "$(ls -A ${QIIME_WRAPPERS})" ]
        then
          for f in `ls ${QIIME_WRAPPERS}/*.xml`;
                do
                        filename=`basename ${f} .xml`
                        echo "Processing of  ${f} file with basename  ${filename}..."
                        sed -i "s|${filename}.py|python2.7.3
                                ${QIIME_SCRIPT_PATH}/${filename}.py|g" ${f}
                done

                for f in `ls ${QIIME_GALAXY_SCRIPT_PATH}/*.py`;
                do
                        echo "Replacing occurences of  ${f} file..."
                        filename=`basename ${f}`
                        filepath=`dirname ${f}`
                        sed -i "s|${filename}|python2.7.3 ${filepath}/${filename}|g"
                                ${QIIME_WRAPPERS}/*
                done

        else
          echo "QIIME-GALAXY Integration failed. Cannot modify wrappers";
        fi
}


##Backup function before installing qiime
backUp() {
```

```
        echo "Inside backUp function"
        echo 'taking backup of important files'
        if [ ! -d "${TEMP_DIR}" ]; then
                mkdir -p ${TEMP_DIR}
        fi
        cp $(12)/.bashrc ${TEMP_DIR}/
        cp ${GALAXY_DIR}/tool_conf.xml ${GALAXY_DIR}/tool_conf.xml.qiime.bkp

}

##Rollback function if qiime installation failed
rollBack() {
        echo "Inside rollBack function"
        echo 'rolling back !!!!!'
        cp ${TEMP_DIR}/.bashrc $(12)/.bashrc
        rm -rf ${QIIME_INSTALL_DIR}
        rm -rf ${R_LIB_PATH}
        cp ${GALAXY_DIR}/tool_conf.xml.qiime.bkp ${GALAXY_DIR}/tool_conf.xml

        source $HOME/.bashrc
}

##Rollback function if qiime installation failed
check_qiime_install() {

        echo "Inside check_qiime_install function"
        line_no=`sed -n "/Packages failed to deploy:/=" ${QIIME_DEPLOY_LOG}`
        failed_tools=`sed "$(($line_no+1)) q;d" ${QIIME_DEPLOY_LOG}`
        if [ "${failed_tools}" ];
        then
                echo 'Qiime Installation failed'
                echo 'Following dependent tools are failed during QIIME installation'
                echo ${failed_tools}
                retval=0
        else
                echo 'Qiime Installed successfully'
                retval=1
        fi
        return ${retval}
}

## Removes unnecessary file after successfull qiime installation
remove_unwanted_files(){

        echo "Inside remove_unwanted_files function"

        echo "removing qiime-deploy-conf directory"
        rm -rf ${QIIME_INSTALL_DIR}/qiime-deploy-conf
        echo "removing qiime-deploy directory"
        rm -rf ${QIIME_INSTALL_DIR}/qiime-deploy
        echo "removing temp directory"
        rm -rf ${TEMP_DIR}

}

## calling backup function
backUp

#calling install_qiime function to install qiime tool
install_qiime

if [ "${1}" = "i" ]
then
        ## Confirmation of installation process
        while true; do
                echo 'Please select the below option to continue'
                echo '1. QIIME INSTALLED SUCCESSFULLY. CONTINUE INTEGRATING QIIME
AND GALAXY.'
                echo '2. QIIME INSTALLED SUCCESSFULLY. EXIT NOW !!'
                echo '3. QIIME INSTALLATION FAILED: REINSTALL QIIME'
                echo '4. QIIME INSTALLATION FAILED. ROLLBACK ANE EXIT !!!'
```

```
                read -p "enter the option here -->" yn
                case $yn in
                        [1]* )
                                echo 'QIIME INSTALLED SUCCESSFULLY. CONTINUE
                                        INTEGRATING QIIME AND GALAXY.';
                                create_python_links
                                modify_activate_sh
                                integrate_on_galaxy
                                modify_wrappers
                                echo 'Integrated QIIME to GALAXY successfully';
                                install_qiime_connector
                                echo 'Qiime connecor installed successfully';
                                break;;
                        [2]* )
                                echo 'QIIME INSTALLED SUCCESSFULLY. EXIT NOW !!';
                                modify_activate_sh
                                create_python_links
                                echo 'EXITING NOW';
                                break;;
                        [3]* )
                                echo 'QIIME INSTALLATION FAILED: REINSTALL QIIME';
                                rollBack
                                install_qiime
                                continue;;
                        [4]* )
                                echo 'QIIME INSTALLATION FAILED. ROLLBACK ANE
                                        EXIT !!!';
                                rollBack
                                break;;
                        * ) echo "Please enter valid option";;
                esac
        done
else
        check_qiime_install
        retval=$?
        echo "returnValue from check Qiime Install:${retval}"
        if [ "${retval}" -eq 0 ]
        then
                echo "Qiime installation failed"
                echo "rolling back and exit ....."
                rollBack
        else
                echo "Qiime installed successfully"
                modify_activate_sh
                create_python_links
                echo "Integrating Qiime and Galaxy....."
                integrate_on_galaxy
                echo 'Integrated QIIME to GALAXY successfully';
                modify_wrappers
                echo 'Qiime wrappers modified successfully';
                install_qiime_connector
                echo 'Qiime connecor installed successfully';
        fi
fi
remove_unwanted_files
```

## 7. REFERENCES


1. Blankenberg D, Von Kuster G, Coraor N, Ananda G, Lazarus R, Mangan M, et al. Galaxy: a web-based genome analysis tool for experimentalists. Current protocols in molecular biology / edited by Frederick M Ausubel [et al]. 2010;Chapter 19:Unit 19.0.1-21.
2. LMU. Corporate Design Manual, LMU. available at http://wwwuni-muenchende/aktuelles/publikationen/cd/download/lmu_cd_manual_1pdf.
3. Corporate Design Manual, Klinikum. available at http://wwwklinikumuni-muenchende/download/de/pressestelle/CD/manualpdf.
4. Kuczynski J, Stombaugh J, Walters WA, Gonzalez A, Caporaso JG, Knight R. Using QIIME to analyze 16S rRNA gene sequences from microbial communities. Current protocols in bioinformatics / editoral board, Andreas D Baxevanis [et al]. 2011;Chapter 10:Unit 10.7.
5. Caporaso JG, Kuczynski J, Stombaugh J, Bittinger K, Bushman FD, Costello EK, et al. QIIME allows analysis of high-throughput community sequencing data. Nature methods. 2010;7(5):335-6.
6. Sun J, Zhang Q, Zhou J, Wei Q. Illumina Amplicon Sequencing of 16S rRNA Tag Reveals Bacterial Community Development in the Rhizosphere of Apple Nurseries at a Replant Disease Site and a New Planting Site. PloS one. 2014;9(10):e111744.
7. Pitta DW, Parmar N, Patel AK, Indugu N, Kumar S, Prajapathi KB, et al. Bacterial diversity dynamics associated with different diets and different primer pairs in the rumen of kankrej cattle. PloS one. 2014;9(11):e111710.
8. Navas-Molina JA, Peralta-Sanchez JM, Gonzalez A, McMurdie PJ, Vazquez-Baeza Y, Xu Z, et al. Advancing our understanding of the human microbiome using QIIME. Methods in enzymology. 2013;531:371-444.
9. phuseman. qiime-deploy. software available at https://githubcom/phuseman/qiime-deploygit. (qiime-deploy).
10. Phuseman. qiime-deploy-conf. software available at https://githubcom/qiime/qiime-deploy-conf.
11. josenavas. qiime-galaxy. available at https://githubcom/qiime/qiime-galaxy.
12. Santaguida PL, Hawker GA, Hudak PL, Glazier R, Mahomed NN, Kreder HJ, et al. Patient characteristics affecting the prognosis of total hip and knee joint arthroplasty: a systematic review. Canadian journal of surgery Journal canadien de chirurgie. 2008;51(6):428-36.