Publish fastq manual

Texas Biomedical Research Institute

publish-fq.sh: script to publish fastq files in structured folders

Author:
Frédéric CHEVALIER

Version 1.0

July 26, 2017

Version history

Version	Date	Modification summary
1.0	2017-05-03	New functionalities described
		Updates from Roy's input
		Spelling and grammar corrections
0.0	2017-02-19	Document created

Contents

Introduction

This script aims at reorganizing fastq files generated by the Illumina bcl2fastq tool in a standardized fashion with a directory named after the flow cell ID containing sub-directories sorted either by lanes/samples or by projects. This script comes in replacement of Juan Peralta's scripts which were used to perform the same reorganization on files generated by the Illumina Casava pipeline. But Illumina is pushing for a switch from the old Casava pipeline to the standalone bcl2fastq program. Because bcl2fastq does not organize the data the same way the pipeline did, Juan's scripts are no longer working. Therefore a complete rewriting was needed. Using this opportunity, the script brings also two new functionalities: sending emails to designated persons when data are available and keeping sequences from undetermined indices.

The bcl2fastq tool demultiplexes and converts raw data from bcl files (acquisition files) to fastq files. This tool comes in replacement of the no longer supported Casava pipeline.[1] This pipeline used to export data by storing fastq files in project and samples directories. Once the Casava output folder generated, Juan's scripts were used to create a folder named after the flow cell ID and then to publish the data in this folder by reorganizing the Casava output based on lanes and samples. Some users wrote scripts based on this folder structure to process data. But the output folder generated by bcl2fastq is much simpler than the one generated by the Casava pipeline: all data are stored in a unique directory or in sub-directories corresponding to each project specified in the sample sheet. The present script aims to recreate the publication style generated by Juan's scripts.

This documentation details the different options available from this script and basic usage examples. Source code of the script can be found at the end as well as on github.

[1] `https://support.illumina.com/sequencing/sequencing_software/casava.html`

Installation and usage

1. Installation

The publish-fq.sh script comes with an optional email template. The script can be located anywhere on the server use to publish the fastq files. For the email option to work, both script and template need to be at the same location, ideally in a folder. The script needs to be executable: the file permission should be `rwxr--r--`. If it is not the case, the command `chmod u+rwx,go+r publish-fq.sh` needs to be executed at the location of the script. For accessing the script from everywhere, the folder path of the script can be appended to the PATH environment variable. To do so in a bash environment, be at the location of the script, execute the command `pwd`, copy the path, open the $HOME/.bashrc file and paste `export PATH=$PATH:script_folder` where `script_folder` is the path given by `pwd`. If you use another shell than bash, this procedure has to be adapted.

2. Usage

Once the publish-fq.sh installed, a usage message is accessible by executing the script without arguments or with -h or --help arguments. The usage message details all options available:

```
    publish-fq.sh -d|--dir bcl2fq_dir -s|--ss samplesheet -o|--dest destination_dir -a|--app-fid string -
        k|--keep-und -p|--pjt -l|--spl -e|--em user_list -h|--help

Aim: Publish fastq files the "old Casava way" (by samples or by projects).

Version: 1.0

Options:
   -d, --dir     path to the output directory of bcl2fq containing the fastq files, the reports and
       stats folders
   -s, --ss      path to the samplesheet
   -o, --dest    path to the destination directory [default: MYPATH]
   -a, --app-fid text appended to the flow cell ID
   -k, --keep-und keep undetermined indices [default: no]
                     If the "Keep undetermined" field is present in the Header section of the sample
                         sheet, this option is ignored
   -p, --pjt     by project publishing [incompatible with -l]
   -l, --spl     by sample publishing [incompatible with -p] [default]
   -e, --em      list of user to which send an email when publishing is done
                     The list must be space separated (eg, -e brad janet) [default: janet]
                     If the "Emails" field is present in the Header section of the sample sheet, this
                         option is ignored
   -h, --help    this message
```

Here are the details of the options:

`--dir:` This option is mandatory. The path to the bcl2fastq output folder must be given. This path can point anywhere on the server.

`--ss:` This option is mandatory. The path to the sample sheet used with bcl2fastq must be given. This path can point anywhere on the server.

`--dest:` The path to the destination (output) folder where the flow cell ID folder will be created to receive the published data in order to mimic the "old Casava way". This path can point anywhere on the server but the HiSeq folder is set by default. This folder is located on the Solexa server.

`--app-fid:` Suffix to add to the flow cell ID folder. This can be useful when two independent publishing from the same flow cell needs to be done (publishing data from demultiplexing of 6 and 8 bp indices for instance). The suffix is automatically prepended with an underscore (e.g., FID_suffix).

`--keep-und:` This option acts as a switch to copy the fastq files containing sequences with undetermined indices. These files are usually not kept because not used in downstream analysis. But if one wants them, the switch can be used to store them in the publish folder. This option can also be set directly in the sample sheet by adding a line in the Header section (see below). If the latter is used, the present option have no effect.

`--pjt:` This option acts as a switch to publish the data following the project names present in the sample sheet. Cannot be used with `--spl`.

`--spl:` This option acts as a switch to publish the data following the lanes and sample names present in the sample sheet. Cannot be used with `--pjt`.

`--em:` This option allows designated correspondent to be informed by email at the end of the publishing. Roy Garcia is informed by default but other people can be included. The email addresses must be separated by a white space. If this is a Texas Biomed address, there is no need to append the @txbiomed.org domain. Email addresses outside of txbiomed.org domain works but the domain needs to be present. If the script does not find any email template, no email is sent and an error message is issued. This option can also be set directly in the sample sheet by adding a line in the Header section (see below). If the latter is used, the present option have no effect.

3. Operation

When a sequencing run is done and bcl2fastq has converted the bcl files into fastq files, the publish script can be called to copy and reorganize the data on the Solexa server. During the process, several explicit messages will appear:

- info message in green about current process,
- warning message in yellow about unexpected but not fatal problem,

- error message in red about unexpected and fatal problem.

When process is done, the final step consists in sending an email at least to the operator (Roy Garcia) and optionally to any designated correspondents. The email address(es) can be added in the sample sheet (recommended) or enter manually by the operator. This email informs correspondent(s) on where the data are stored and to what project(s) the data are related, and send the sequencing report to these persons. The report will be compressed in tar.gz file. This report archive will actually contains the Reports and the Stats folders generated by bcl2fastq. Presence of each folder will be check and warning message will be issued specifically if they are missing. To work properly, a file named `email_template.txt` needs to be present along the script. If not, no email will be sent despite the activation of the option.
N.B.: If publish-fastq.sh is interrupted in the middle of the process, data will be corrupted. In this case, the published folder must be deleted and the publishing restarted.

4. Usage examples

Here is an example of how to use the script:

```
# In the case publish-fq.sh can be reached through PATH
# The following command will publish data contained in the shsmchronobio using the
# sample sheet present in the folder and sending to the corresponding author
publish-fq.sh -d shsmchronobio -s shsmchronobio/samplesheet.csv -e brad

# In the case publish-fq.sh cannot be reached through PATH
# The following example does the same as previously. Notice the change at the beginning:
# path/to/publish-fq/ is the absolute path of the folder containing publish-fq.sh
/path/to/publish-fq/publish-fq.sh -d shsmchronobio -s shsmchronobio/samplesheet.csv -e brad
# or
cd /path/to/publish-fq/
./publish-fq.sh -d shsmchronobio -s shsmchronobio/samplesheet.csv -e brad

# If a new publishing needs to be performed after a first publishing, this option -a must be used.
# Here an example where the "new_publishing" suffix will be appended to the directory name
# (i.e., the flow cell ID)
cd /path/to/publish-fq/
./publish-fq.sh -d shsmchronobio -s shsmchronobio/samplesheet.csv -a new_publishing -e brad
```

Special fields of the sample sheet

Two new fields can now be added to the Header section of the sample sheet. These fields are not used by bcl2fastq and does not disturbed the demultiplexing. They allow you to interact with two options related to the publishing step: sending emails and keeping undetermined indices sequences. They are highly recommended in order to avoid misunderstanding between users of the sequencing facility and the operator (Roy). For those using the IEM software from Illumina to generate the sample sheet, these fields have to be added manually by opening the file in a spreadsheet program (Excel, Calc, ...) or in a text editor (Notepad, Geany, ...).

1. Emails

A field named "Emails" (or "Email" or "emails" or "email") can be added to trigger the email option of the publishing script. This method is recommended over asking the operator to add email addresses when publishing data on the ranch. Recipients of the email can be the PI, the research assistant who prepared the libraries and the bioinformatician who will process the data for instance.

This option accepts a list of email addresses separated by white space. Texas Biomed email addresses does not need to have their domain (@txbiomed.org) appended, only the user name is needed. For instance, "brad" is enough to send an email to "fcheval@txbiomed.org". Email addresses outside the Texas Biomed domain can be used but in this case the domain (@something.edu) must be present. For an example, see Sample sheet example

The email option triggered through the sample sheet has priority over the email arguments from command line. Therefore one cannot fill the sample sheet then ask the operator to add an email address with the command line. In this case the sample sheet needs to be modified.

2. Undetermined indices

Sequences with undetermined indices are generated during the demultiplexing process when bcl2fastq cannot identify the index sequence with certainty. By default bcl2fastq allows one mismatch in the index sequence. This undetermined sequences are usually discarded because unassigned. But if for some reason one wants to keep and analyze them, they can be retained when publishing the data and kept in a dedicated folder within the flow cell ID folder.

To keep the undetermined sequences, a field named "Keep undetermined" can be created in the Header section of the sample sheet the same way as the email field. This field accepts as values yes (or y) and no (or n). By default if the field is absent and if the switch is not used in command line, these sequences are discarded. They can only be regenerated by using bcl2fastq again with the same options as for the first demultiplexing. For a usage example, see Sample sheet example

As for the email option, the keep option triggered through the sample sheet has priority over the switch from command line.

3. Sample sheet example

Here is an example of the Header section of the sample sheete with the two new fields at the end:

```
[Header]
IEMFileVersion       4
Investigator Name    Tim Anderson
Experiment Name      Transmission
Date                 4/17/2017
Workflow             GenerateFASTQ
Application          HiSeq FASTQ Only
Assay                TruSeq LT
Description
Chemistry            Default
Emails               brad mary pierre xyz@uthscsa.edu
Keep undetermined    no
```

N.B.: The sample sheet is a comma separated values file. Therefore the presentation above reflects what one can see in a spreadsheet editor but not in the text editor.

Source code

1. Script code

This is the code of the publish-fq.sh script. The version of the script is indicated within the code and details about version history is given in the corresponding section of the script.

```bash
#!/bin/bash
# Title: publish-fq.sh
# Version: 1.1
# Author: Frédéric CHEVALIER <fcheval@txbiomed.org>
# Created in: 2016-11-05
# Modified in: 2017-06-02
# License : GPL v3



#======#
# Aims #
#======#

aim="Publish fastq files the \"old Casava way\" (by samples or by projects)."



#==========#
# Versions #
#==========#

# v1.1 - 2017-06-02: bug corrected when Sample_ID and Sample_name are different
# v1.0 - 2017-04-30: safe script exit on any command errors / optional suffix to flow cell ID added /
    samplesheet reading improved because of DOS formatting / emailing step improved by sending email at
    once to all recipients and by adding project list / cleaning step updated
# v0.1 - 2017-04-18: flow cell ID automatically set / samplesheet file copy / auto clean
# v0.0 - 2016-11-05: creation

version=$(grep -i -m 1 "version" "$0" | cut -d ":" -f 2 | sed "s/^ *//g")



#===========#
# Functions #
#===========#

# Usage message
function usage {
    echo -e "
    \e[32m ${0##*/} \e[00m -d|--dir bcl2fq_dir -s|--ss samplesheet -o|--dest destination_dir -a|--app-fid
        string -k|--keep-und -p|--pjt -l|--spl -e|--em user_list -h|--help
```

```
Aim: $aim

Version: $version

Options:
    -d, --dir      path to the output directory of bcl2fq containing the fastq files, the reports and
        stats folders
    -s, --ss       path to the samplesheet
    -o, --dest     path to the destination directory [default: MYPATH]
    -a, --app-fid  text appended to the flow cell ID
    -k, --keep-und keep undetermined indices [default: no]
                       If the \"Keep undetermined\" field is present in the Data section of the sample
                           sheet, this option is ignored
    -p, --pjt      by project publishing \e[31m[Incompatible with -l]\e[00m
    -l, --spl      by sample publishing \e[31m[Incompatible with -p]\e[00m [default]
    -e, --em       list of user to which send an email when publishing is done
                       The list must be space separated (eg, -e brad janet) [default: janet]
                       If the \"Emails\" field is present in the Data section of the sample sheet, this
                           option is ignored
    -h, --help     this message
    "
}


# Info message
function info {
    if [[ -t 1 ]]
    then
        echo -e "\e[32mInfo:\e[00m $1"
    else
        echo -e "Info: $1"
    fi
}


# Warning message
function warning {
    if [[ -t 1 ]]
    then
        echo -e "\e[33mWarning:\e[00m $1"
    else
        echo -e "Warning: $1"
    fi
}


# Error message
## usage: error "message" exit_code
## exit code optional (no exit allowing downstream steps)
function error {
    if [[ -t 1 ]]
    then
        echo -e "\e[31mError:\e[00m $1"
    else
        echo -e "Error: $1"
    fi

    if [[ -n $2 ]]
    then
        exit $2
    fi
}


# Dependency test
function test_dep {
    which $1 &> /dev/null
    if [[ $? != 0 ]]
    then
        error "Package $1 is needed. Exiting..." 1
    fi
}


# Progress bar
## Usage: ProgressBar $mystep $myend
function ProgressBar {
    if [[ -t 1 ]]
    then
        # Process data
        let _progress=(${1}*100/${2}*100)/100
        let _done=(${_progress}*4)/10
        let _left=40-$_done
        # Build progressbar string lengths
        _fill=$(printf "%${_done}s")
        _empty=$(printf "%${_left}s")

        # Build progressbar strings and print the ProgressBar line
        # Output example:
        # Progress : [########################################] 100%
        printf "\r\e[32mProgress:\e[00m [${_fill// /=}${_empty// / }] ${_progress}%%"

        [[ ${_progress} == 100 ]] && echo ""
    fi
}


# Clean up function for trap command
## Usage: clean_up file1 file2 ...
function clean_up {
    rm -rf $@
    exit 1
}



#==============#
# Dependencies #
#==============#

test_dep perl



#===========#
# Variables #
#===========#

# Options
while [[ $# -gt 0 ]]
do
```

```
    case $1 in
        -d|--dir     ) dir_fq="$2"    ; shift 2 ;;
        -s|--ss      ) samplesheet="$2" ; shift 2 ;;
        -o|--dest    ) dir_out="${2}" ; shift 2 ;;
        -a|--app-fid ) txt_app="${2#_}" ; shift 2 ;;  # Clean txt_app if "_" already prepended
        -k|--keep-und ) keep=keep     ; shift  ;;
        -p|--pjt     ) pjt="project"  ; shift  ;;
        -l|--spl     ) spl="sampple"  ; shift  ;;
        -e|--em      ) myemails="$2"  ; shift 2
                        while [[ ! -z "$1" && $(echo "$1"\ | grep -qv "^-" ; echo $?) == 0 ]]
                        do
                            myemails="$myemails $1"
                            shift
                        done ;;
        -h|--help    ) usage ; exit 0 ;;
        *            ) error "Invalid option: $1\n$(usage)" 1 ;;
    esac
done


# Check the existence of obligatory options
if [[ -z "$dir_fq" ]]
then
    error "The -d option is required. Exiting...\n$(usage)" 1
elif [[ -z "$samplesheet" ]]
then
    error "The -s option is required. Exiting...\n$(usage)" 1
fi

if [[ -n $pjt && -n $spl ]]
then
    error "The -p and -l options cannot be used at the same time. Exiting..." 1
elif [[ -z $pjt && -z $spl ]]
then
    spl="sample"
fi

[[ -n $spl ]] && myout_type="sample" || myout_type="project"


# Default value for output directory
[[ -z "$dir_out" ]] && dir_out=MYPATH


# Operator email address
operator=janet@txbiomed.org

# Email template
email_tmpl="$(dirname "$0")/email_template.txt"


# Initiate counter
count=1



#============#
# Processing #
#============#

# Stop script on any command errors or non declared variables
```

```
set -e

#info "The fastq files will be published by $myout_type."

# List fastq files
list_fq=$(find "$dir_fq" -type f -name *.fastq*)

# Get flowcell ID
FC_ID=$(grep -m 1 -i "flowcell" "$dir_fq/Reports/html/index.html" | perl -pe "s,.*src=.(.*?)/.*,\1,")

if [[ -z $FC_ID ]]
then
    error "Flowcell ID not set. Is the Reports directory present? Exiting..." 1
else
    info "Flowcell ID $FC_ID"
fi

# Update flow cell ID if needed
[[ -n "$txt_app" ]] && FC_ID="${FC_ID}_${txt_app}"

# Get and check dir_out
dir_out="${dir_out%%/}/${FC_ID}"
[[ -d "$dir_out" ]] && error "Output directory $dir_out exists already. You should use -a option if you
     want to do a new publishing. Exiting... " 1


#------------------#
# Samplesheet info #
#------------------#

# Reformat any carriage return character of the samplesheet
samplesheet_data=$(sed 's/\r/\n/g' "$samplesheet")

# Extract header section
header_sec=$(echo "$samplesheet_data" | sed -n "/\[Header\]/,/\[/{//d;p}")

# Identify the first line of the [Data] section
data_ln=$(echo "$samplesheet_data" | grep -in "^\[data\]" | cut -d ":" -f 1)

# Extract the data section
data_sec=$(echo "$samplesheet_data" | tail -n +${data_ln})

# Extract the header line
data_head_ln=$(echo "$data_sec" | grep -ni "index" | cut -d ":" -f 1)

# Extract index and lane column number
index_cln=$(echo "$data_sec" | sed -n "${data_head_ln}p" | sed "s/,/\n/g" | grep -ni "^index$" | cut -d "
     :" -f 1)
lane_cln=$(echo "$data_sec" | sed -n "${data_head_ln}p" | sed "s/,/\n/g" | grep -ni "lane" | cut -d ":" -
     f 1)
sample_cln=$(echo "$data_sec" | sed -n "${data_head_ln}p" | sed "s/,/\n/g" | grep -ni "sample_id" | cut -
     d ":" -f 1)
project_cln=$(echo "$data_sec" | sed -n "${data_head_ln}p" | sed "s/,/\n/g" | grep -ni "project" | cut -d
      ":" -f 1)

# Check for uniqueness of each column
[[ $(echo $index_cln | tr " " "\n" | wc -l) != 1 ]] && error "Index column cannot be identified properly
     (none or more than one). Exiting..." 1
[[ $(echo $lane_cln | tr " " "\n" | wc -l) != 1 ]] && error "Lane column cannot be identified properly (
     none or more than one). Exiting..." 1
[[ $(echo $sample_cln | tr " " "\n" | wc -l) != 1 ]] && error "Sample_ID column cannot be identified
```

```
    properly (none or more than one). Exiting..." 1
[[ $(echo $project_cln | tr " " "\n" | wc -l) != 1 ]] && error "Project column cannot be identified
    properly (none or more than one). Exiting..." 1

# Refine data section to remove header lines
((data_head_ln++))
data_sec=$(echo "$data_sec" | tail -n +${data_head_ln})



#----------------#
# By lane/sample #
#----------------#

if [[ -n $spl ]]
then

    # Get sample list
    mysamples=$(echo "$data_sec" | cut -d "," -f $sample_cln | sort | uniq | sed "/^$/d")
    mysamples_lg=$(echo "$mysamples" | wc -l)

    # Treat each sample
    for i in $mysamples
    do
        ProgressBar $count $mysamples_lg ||:

        # Get all lines corresponding to the sample (if split in several lanes for instance)
        sample_sec=$(echo "$data_sec" | awk -v sample_cln=$sample_cln -v i=$i -F "," '$sample_cln == i {
            print $0}')

        for j in $(echo "$sample_sec" | cut -d "," -f $lane_cln)
        do
            # Create dir_out/Lane_sample
            mkdir -p "$dir_out/Lane${j}_${i}"

            # Generate list of fastq files
            if [[ -n $(echo "$list_fq" | grep "${i}_.*_L00${j}") ]] # When Sample_ID and Sample_Name are
                the same
            then
                myfiles=$(echo "$list_fq" | grep "${i}_.*_L00${j}")
            elif [[ -n $(echo "$list_fq" | grep "/${i}/") ]]      # When Sample_ID and Sample_Name are
                different, look at folder with Sample_ID
            then
                myfiles=$(echo "$list_fq" | grep "/${i}/")
            else
                error "fastq.gz files from $i sample are expected but missing. Exiting..." 1
            fi

            # Copy any fastq corresponding to lane and sample in this directory (for test do touch only)
            cp -t "$dir_out/Lane${j}_${i}" $myfiles
        done

        ((count++))
    done

    # Copy sample sheet in the output directory
    cp -a "$samplesheet" "$dir_out/$(basename "$samplesheet")"

fi


#------------#
```

```
# By project #
#------------#

if [[ -n $pjt ]]
then

    # Get index list
    myindexes=$(echo "$data_sec" | cut -d "," -f $index_cln | sort | uniq | sed "/^$/d")
    myindexes_lg=$(echo "$myindexes" | wc -l)

    # Treat each index (ie, Project)
    for i in $myindexes
    do
        ProgressBar $count $myindexes_lg ||:

        # Get all lines corresponding to the indexes (if split in several lanes for instance)
        sample_sec=$(echo "$data_sec" | awk -v index_cln=$index_cln -v i=$i -F "," '$index_cln == i {print
             $0}')

        for j in $(echo "$sample_sec" | cut -d "," -f $sample_cln)
        do
            # Create dir_out/Lane_sample
            mkdir -p "$dir_out/Project_${count}/Sample_${j}"

            # List lanes corresponding to the sample
            lane_lst=$(echo "$sample_sec" | awk -v sample_cln=$sample_cln -v lane_cln=$lane_cln -v j=$j -F
                 "," '$sample_cln == j {print $lane_cln}')

            for k in $lane_lst
            do
                # Generate list of fastq files
                if [[ -n $(echo "$list_fq" | grep "${j}_.*_L00${k}") ]] # When Sample_ID and Sample_Name
                     are the same
                then
                    myfiles=$(echo "$list_fq" | grep "${j}_.*_L00${k}")
                elif [[ -n $(echo "$list_fq" | grep "/${j}/") ]]      # When Sample_ID and Sample_Name are
                     different, look at folder with Sample_ID
                then
                    myfiles=$(echo "$list_fq" | grep "/${j}/")
                else
                    error "fastq.gz files from $j sample are expected but missing. Exiting..." 1
                fi

                # Copy any fastq corresponding to lane and sample in this directory (for test do touch only
                     )
                cp -t "$dir_out/Project_${count}/Sample_${j}" $myfiles
            done
        done

        ((count++))
    done

    # Copy sample sheet in the output directory
    cp -a "$samplesheet" "$dir_out/$(basename "$samplesheet")"

fi


#---------------------------#
# Keep undetermined indices #
#---------------------------#
```

```
# Look for keeping undetermined indices from the samplesheet
undetermined_ln=$(echo "$header_sec" | egrep -i "^keep undetermined," | cut -d "," -f 2)

[[ -n $keep && -n "$undetermined_ln" ]] && warning "\"Keep undetermined\" present in the command line and
     in the samplesheet. Only option from the samplesheet will be used."

# Check the variable (yes or no, case insensitive)
if  [[ -n "$undetermined_ln" && $undetermined_ln =~ ^[yY][eE]?[sS]?$ ]]
then
    keep=keep
elif [[ -n "$undetermined_ln" && $undetermined_ln =~ ^[nN][oO]?$ ]]
then
    keep=
elif [[ -n "$undetermined_ln" && ! ($undetermined_ln =~ ^[yY][eE]?[sS]?$ || $undetermined_ln =~ ^[nN][oO
    ]?$) ]]
then
    warning "\"Keep undetermined\" identified in the samplesheet did not match yes or no (current value:
        $undetermined_ln). Skipping..."
    keep=
    error="keep undetermined"
fi

# Keep undetermined in a dedicated folder if
if [[ -n $keep ]]
then
    # Create undetermined indices directory
    mkdir "$dir_out/Undetermined_indices"

    # Copy the fastq file of undermined indices
    cp -a "$dir_fq"/Undetermined_*.fastq.gz "$dir_out/Undetermined_indices"
fi


#--------------------------#
# Export Reports and Stats #
#--------------------------#

if [[ ! -e "$dir_fq/Reports" ]]
then
    warning "Reports folder not found."
elif [[ ! -e "$dir_fq/Stats" ]]
then
    warning "Stats folder not found."
else
    tar -czf "$dir_out/Reports & Stats.tar.gz" -C "$dir_fq" "Reports" "Stats"
fi


#-------#
# Email #
#-------#

# Get email addresses from the samplesheet if any
myemails_ln=$(echo "$header_sec" | egrep -i "^e-?mails?," | cut -d "," -f 2)

# Check if emails are present in the command line and the sample sheet
if [[ -n "$myemails" && -n "$myemails_ln" ]]
then
    warning "Email addresses identified in the command line and the samplesheet. Only email address(es)
        from the samplesheet will be used."
    myemails="$myemails_ln"
fi

# Append txbiomed domain for email addresses if needed
if [[ -n "$myemails" ]]
then
    for i in $myemails
    do
        if [[ ! $(echo "$i" | grep "@") ]]
        then
            j=${i}@txbiomed.org
        fi

        # Update email address list
        myemails=$(echo $myemails | sed "s/$i/$j/")
    done
fi

# Prepare email
if [[ -n "$myemails" && ! -s "$email_tmpl" ]]
then
    error "Email template $email_tmpl does not exist or is empty. Skipping email step..."
elif [[ -z "$myemails" ]]
then
    echo -e "Dear ${operator%@*}\n\nData were uploaded at $dir_out but no email address of corresponding
        author(s) was found. So no email was sent." | mail -s "Sequencing data available - no email sent"
        $operator
elif [[ -n "$myemails" ]]
then
    # Format list of recipient(s)
    i=$(echo $myemails | sed -r "s/@[[:alnum:].]* /, /g ; s/@[[:alnum:].]*$// ; s/(.*),/\1 and/")

    # Create and format project list
    myprojects=$(echo "$data_sec" | cut -d "," -f $project_cln | sort | uniq | sed -r "/^$/d ; s/^/ - /" |
         sed "1 s/^/\\n/")

    # If no project list, default value
    [[ -z "$myprojects" ]] && mypjts="none."

    # Send email
    eval "cat <<< \"$(<"$email_tmpl")\"" | mail -s "Sequencing data available" -a "$dir_out/Reports &
        Stats.tar.gz" -c $operator $myemails
fi


#----------#
# Clean up #
#----------#

if [[ -n $error ]]
then
    warning "Error found with \"$error\". Skipping cleaning step..."
else
    clean_up "$dir_fq"
fi


exit 0
```

2. Email template

This is the email_template.txt called by the script in order to send an email when the publishing is done. The body of the template can be adjusted according to any specific needs. This template contains three variables:

- `${i%%@*}`: contains the email user names of the recipients,
- `$dir_out`: contains the path to the published data,
- `$myprojects`: contains the list of projects founds in the sample sheet. If there is no project, the default value is none.

```
Dear ${i%%@*}

Your sequencing data are available on the ranch at the following location: \""$dir_out"\".
This sequencing run is related to the following project(s): $myprojects

Sequencing report from bcl2fq is also attached. To see the report, uncompress the tar.gz archive and open
     the index.html file present in Reports/html.

Should you have any questions, please let me know.

Roy GARCIA
```