datamanager Documentation

Release 1.0

William Digan Bastien Rance Hector Countouris

Nov 18, 2016

CONTENTS

Contents:

CHAPTER

ONE

DATAMANAGERPKG PACKAGE

1.1 datamanagerpkg Package

datamanagerpkg.__init__.**grou**()

1.2 GalaxyCommunication_data_manager Module

This module illustrates how to write GalaxyCommunication_data_manager.pyc and ProtonCommunication_data_manager.py Basically it is just a sphinx test for the documentation

datamanagerpkg.GalaxyCommunication_data_manager.**CNV_Input_Dict**(*galaxyWeb*, *historyID*)

returns (data_Input_CNVID)

Descriptions:

This function return a dictionnary whitch contains datasets id for CNV input files. This dictionnary contains a bcsummary and bcmatrix keys.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **historyID** (*string*) – a galaxy history ID

Returns data_Input_CNVID

Return type dictionnary

datamanagerpkg.GalaxyCommunication_data_manager.**Create_History**(*galaxyWeb*, *workflow_Name*)

returns (historyDict)

Descriptions:

This function create a new galaxy history where the data will be load.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **workflow_Name** (*string*) – part of the name of the history

Returns historyDict

Return type dict

datamanagerpkg.GalaxyCommunication_data_manager.**Run_CNV_Workflow**(*galaxyWeb*, *data_Input_CNVID*, *historyID*)

returns (int)

Descriptions:

This function retrieve the CNV workflow and execute it. Use a dictionnary as input.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **data_Input_CNVID** (*dictionnary*) – a dictionnary output from function CNV_Input_Dict
- **historyID** (*string*) – a galaxy history ID

Returns 1

Return type int

datamanagerpkg.GalaxyCommunication_data_manager.**addAllWorkflow**(*galaxyWeb*, *workflow_Dir*)

returns (int)

Descriptions:

This function aims to load all workflows on a folder such as '/nas_Dir/workflow' for the current users.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **workflow_Dir** (*string*) – path to the workflow directory

Returns 0 or 1

Return type int

Note: This function need to be used only one time when the

Galaxy user api key is generated

datamanagerpkg.GalaxyCommunication_data_manager.**createUserApikey**(*galaxyWeb*, *userID*)

returns (userApiKey)

Descriptions:

This function aims to return the galaxy users dictionnary.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **userID** (*string*) – the current user ID in Galaxy

Returns userApiKey

Return type string

Note: In this function I can not use the users.get_current_user() function from bioblend because I use the Galaxy Master ApiKey

---

datamanagerpkg.GalaxyCommunication_data_manager.**galaxyConnection**(*base_url*, *apiKey*)

returns (GalaxyInstance)

Descriptions:

This function aims to create a connection to the Galaxy server.

Parameters:

Parameters

- **base_url** (*string*) – an url which point to your galaxy instance
- **apiKey** (*string*) – a valid galaxy API key

Returns GalaxyInstance

Return type GalaxyInstance

datamanagerpkg.GalaxyCommunication_data_manager.**mainCNV**(*expDict*, *base_url*, *apiKey*)

returns (historyID)

Descriptions:

This function execute the CNV routine. From a run of the Ion Proton, The routine will connect the user to Galaxy, create an history, upload the CNV input files to it and run the CNV workflow.

Parameters:

Parameters

- **expDict** – a dictionnary output from ProtonCommunication_data_manager.copyData().
- **base_url** (*string*) – an url which point to your galaxy instance
- **apiKey** (*string*) – a valid galaxy API key

Returns historyID the galaxy history where the data and the CNV run are located

Rtype historyID a dictionnary

datamanagerpkg.GalaxyCommunication_data_manager.**returnGalaxyUsers**(*galaxyWeb*)

returns (usersDict)

Descriptions:

This function aims to return the galaxy users dictionnary.

Parameters:

Parameters **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance

Returns usersDict

Return type Dictionnary

Note: In this function I can not use the users.get_current_user() function from bioblend because I use the Galaxy Master ApiKey

datamanagerpkg.GalaxyCommunication_data_manager.**upload_To_History_CNV**(*galaxyWeb*, *expDict*, *historyID*)

returns (int)

Descriptions:

This function upload to a specific history the CNV data.

Parameters:

Parameters

- **galaxyWeb** (*GalaxyInstance*) – a connection to your galaxy instance
- **expDict** (*dictionnary*) – a result dictionnary output from the ProtonCommunication script
- **historyID** (*string*) – a galaxy history ID

Returns 1

Return type int

1.3 Main_data_manager Module

1.4 ProtonCommunication_data_manager Module

The module ProtonCommunication_data_manager.py was designed to be able to connect to the HEGP Ion Proton and copy Data easily. It can be used with GalaxyCommunication_data_manager.py which assure the Data-Manager_Galaxy Job routine.

This script use the The Torrent Suite Software Development Kit to communicate with the Ion Proton. ProtonCommunication_data_manager fullfill three main goals: - retrieve the Data - Select the data you want to use - Copy them throught the network

datamanagerpkg.ProtonCommunication_data_manager.**CheckExperiments**(*nb_limit*, *idpwd*, *base_url*)

returns (dictionnary)

Descriptions:

This function aims to return a dictionnary, which contains the 'n' last experiments. It also return the run status and if the run is Complete or not. For that purpose you need to provide the following parameters.

Parameters:

Parameters

- **nb_limit** (*int*) – the 'n' number of experiments to check out
- **idpwd** (*string*) – the user ID to connect to the proton

- **idpwd** – the user ID to connect to the proton(To add)
- **base_url** (*string*) – the Ion Proton URL

Returns dict

Return type

dict

This function retrieve the n last experiment of the Ion Proton. it returns a dictionnary which contains 5 elements. {RunName: {cnvFileName;status;ftpStatus;date;id;resultsQuery}}

Note: Dictionnary structure: {RunName: {cnvFileName;status;ftpStatus;date;id;resultsQuery}}

- RunName : the experiments run name
- cnvFileName : the sting match for the bcsummary and bcmatrix file
- status : run status either 'pending' or 'run'
- ftpStatus : if the data can be download, either 'Complete' or ''
- date : project date
- id : project id in the ion proton
- resultsQuery: astring to the result folder '

datamanagerpkg.ProtonCommunication_data_manager.**CheckResConsistency**(*expDict*, *ssh*)

returns (dictionnary)

Descriptions:

This function check data consistency for one dictionnary before performed a scp command. Quality control need to be handle in this function. add a key coverageAnalysis_out which point to the right folder coverageAnalysis_out which contains the bed file ColonLungV2.20140523

Parameters:

Parameters

- **expDict** – a dictionnary output from QueryResults()
- **ssh** (sshConnection) – sshConnection from sshConnection() :type expDict: dict

Returns dict

Return type dict

datamanagerpkg.ProtonCommunication_data_manager.**CheckResultsConsistency**(*expDict*, *ssh*)

returns (dictionnary)

Descriptions:

This function check data consistency for a a dict of dictionnary before performed a scp command. Quality control need to be handle in this function. add a key coverageAnalysis_out which point to the right folder coverageAnalysis_out which contains the bed file ColonLungV2.20140523

Parameters:

Parameters

- **expDict** – a dictionnary output from QueryResults()
- **ssh** (sshConnection) – sshConnection from sshConnection() :type expDict: dict

Returns dict

Return type dict

datamanagerpkg.ProtonCommunication_data_manager.**FindResults**(*expDict*, *idpwd*, *base_url*)

returns (dictionnary)

Descriptions:

This function find the result folder absolut path. it retuns a dictionnary and add the filesystempath to the current dictionnary.

Parameters:

Parameters

- **expDict** (*dict*) – a directory output from the the CheckExperiments() function
- **idpwd** (*string*) – the user ID to connect to the proton
- **idpwd** – the user ID to connect to the proton(To add)
- **base_url** (*string*) – the Ion Proton URL

Returns dict

Return type

dict

This function retrieve the result path associated with an experiments name from the Ion Proton.

it returns a dictionnary which contains 2 new elements from the current dictionnary.

{RunName: {resultsName;runPath;...}}

Note: Dictionnary structure: {RunName: {resultsName;runPath;cnvFileName;status;ftpStatus;date;id;resultsQuery}}

- resultsName; : the experiments result name
- runPath : the path to the current result folder in the Ion Proton

datamanagerpkg.ProtonCommunication_data_manager.**QueryResults**(*resultsQuery*, *idpwd*, *base_url*)

returns (dictionnary)

Descriptions:

This function find the result folder absolut path associated to an identified resultsQuery. it returns a dictionnary and add the filesystempath to the current dictionnary.

Parameters:

Parameters

- **resultsQuery** – a string resultsQuery output from CheckExperiments()
- **idpwd** (*string*) – the user ID to connect to the proton
- **idpwd** – the user ID to connect to the proton(To add)
- **base_url** (*string*) – the Ion Proton URL

Returns dict

Return type

dict

This function retrieve the result path associated with an experiments name from the Ion Proton.

it returns a dictionnary which contains 2 new elements from the current dictionnary.

{RunName: {cnvFileName;resultsName;runPath;}}

Note: Dictionnary structure: {RunName: {resultsName;runPath;cnvFileName}}

- resultsName; : the experiments result name
- runPath : the path to the current result folder in the Ion Proton

datamanagerpkg.ProtonCommunication_data_manager.**copyData**(*currentExp*, *ssh*)

returns (dictionnary)

Descriptions:

This function copy data trought scp and perform checksum. Add the key bcmatrix and bcsummary to the current directory. if some rename opperation need to be performed it as to been done here.

Parameters:

Parameters

- **currentExp** (*dict*) – a directory output from CheckResultsConsistency()
- **ssh** (sshConnection) – sshConnection from sshConnection()

Returns dict

Return type dict

datamanagerpkg.ProtonCommunication_data_manager.**mainGetCNVData**(*base_url*, *idpr*, *severName*, *experimentLimit*)

datamanagerpkg.ProtonCommunication_data_manager.**sshConnection**(*severName*, *idpwd*)

returns (sshConnection)

Descriptions:

This function allow an ssh connection through the pakito python module. the goal here is to establish a connection before performed an scp bash command.

Parameters: :param severName: name of the linux machine to connect throught ssh :param idpwd: the user ID to connect to the proton :param idpwd: the user ID to connect to the proton(To add) :type severName: string :type idpwd: string :type idpwd: string :returns: sshConnection :rtype: sshConnection

1.5 addWorkflow Module

datamanagerpkg.addWorkflow.**CNV_Input_Dict**(*galaxyWeb*, *historyID*)

datamanagerpkg.addWorkflow.**Create_History**(*galaxyWeb*, *workflow_Name*)

datamanagerpkg.addWorkflow.**Run_CNV_Workflow**(*galaxyWeb*, *data_Input_CNVID*, *historyID*)

datamanagerpkg.addWorkflow.**addAllWorkflow**(*galaxyWeb*, *workflow_Dir*)

datamanagerpkg.addWorkflow.**mainCNV**(*pathToFile*, *apiKey*, *inputAbsolutPath*)

datamanagerpkg.addWorkflow.**upload_To_History**(*galaxyWeb*, *filesPath*, *historyID*, *inputAbsolutPath*)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

d

U