Python library for running IGF pipelines Documentation

Release 1.5

Avik Datta, IGF Team

Sep 06, 2022

TABLE OF CONTENTS

CHAPTER ONE

LIST OF PYTHON SCRIPTS

1.1 Sequencing run processing

1.1.1 Metadata registration

Usage

find_and_register_project_metdata.py
[-h] -p PROJET_INFO_PATH -d DBCONFIG -t USER_ACCOUNT_TEMPLATE -n SLACK_CONFIG -u HPC_USER -a HPC_ADDRESS -l LDAP_SERVER [-h] [-s] [-c] [-i] [-m]

Parameters

-h, --help	: Show this help message and exit
-p, --projet_info_path	: Project metdata directory path
-d, --dbconfig	: Database configuration file path
-t, --user_account_template	: User account information email template file path
-s, --log_slack	: Toggle slack logging
-n, --slack_config	: Slack configuration file path
-c, --check_hpc_user	: Toggle HPC user checking
-u, --hpc_user	: HPC user name for ldap server checking
-a, --hpc_address	: HPC address for ldap server checking
-l, --ldap_server	: Ldap server address
-i, --setup_irods	: Setup iRODS account for user
-m, --notify_user	: Notify user about new account and password

1.1.2 Monitor sequencing run for demultiplexing

Usage

find_new_seqrun_and_prepare_md5.py
[-h] -p SEQRUN_PATH -m MD5_PATH -d DBCONFIG_PATH -s SLACK_CONFIG -a ASANA_CONFIG -i ASANA_PROJECT_ID -n PIPELINE_NAME -j SAMPLESHEET_JSON_SCHEMA [-e EXCLUDE_PATH]

Parameters

-h, --help	: show this help message and exit
-p, --seqrun_path SEQRUN_PATH	: Seqrun directory path
-m, --md5_path MD5_PATH	: Seqrun md5 output dir

-d, --dbconfig_path DBCONFIG_PATH : Database configuration json file

-s, --slack_config SLACK_CONFIG : Slack configuration json file

-a, --asana_config ASANA_CONFIG : Asana configuration json file

-i, --asana_project_id ASANA_PROJECT_ID : Asana project id

-n, --pipeline_name PIPELINE_NAME : IGF pipeline name

-j, --samplesheet_json_schema SAMPLESHEET_JSON_SCHEMA : JSON schema for samplesheet validation

-e, --exclude_path EXCLUDE_PATH : List of sub directories excluded from the search

1.1.3 Switch off project barcode checking

Usage

mark_project_barcode_check_off.py
[-h] -p PROJET_ID_LIST -d DBCONFIG [-s] -n SLACK_CONFIG

Parameters

-h, --help : show this help message and exit

-p, --projet_id_list PROJET_ID_LIST : A file path listing project_igf_id

-d, --dbconfig DBCONFIG : Database configuration file path

-s, --log_slack : Toggle slack logging

-n, --slack_config SLACK_CONFIG : Slack configuration file path

1.1.4 Accept modified samplesheet for demultiplexing run

Usage

reset_samplesheet_for_pipeline.py
[-h] -p SEQRUN_PATH -d DBCONFIG -n SLACK_CONFIG -a ASANA_CONFIG -i ASANA_PROJECT_ID -f INPUT_LIST

Parameters

-h, --help : show this help message and exit

-p, --seqrun_path SEQRUN_PATH : Sequencing run directory path

-d, --dbconfig DBCONFIG : Database configuration file path

-n, --slack_config SLACK_CONFIG : Slack configuration file path

-a, --asana_config ASANA_CONFIG : Asana configuration file path

-i, --asana_project_id ASANA_PROJECT_ID : Asana project id

-f, --input_list INPUT_LIST : Sequencing run id list file

1.1.5 Copy files to temp directory for demultiplexing run

Usage

moveFilesForDemultiplexing.py
[-h] -i INPUT_DIR -o OUTPUT_DIR -s SAMPLESHEET_FILE -r RUNINFO_FILE

Parameters

-h, --help : show this help message and exit

-i, --input_dir INPUT_DIR : Input files directory

-o, --output_dir OUTPUT_DIR : Output files directory

-s, --samplesheet_file SAMPLESHEET_FILE : Illumina format samplesheet file

-r, --runinfo_file RUNINFO_FILE : Illumina format RunInfo.xml file

1.1.6 Transfer metadata to experiment from sample entries

Usage

update_experiment_metadata_from_sample_attribute.py [-h] -d DBCONFIG -n SLACK_CONFIG

Parameters

-h, --help show this help message and exit

-d, --dbconfig DBCONFIG : Database configuration file path

-n, --slack_config SLACK_CONFIG : Slack configuration file path

1.2 Pipeline control

1.2.1 Reset pipeline for data processing

Usage

batch_modify_pipeline_seed.py [-h] -t TABLE_NAME -p PIPELINE_NAME
-s SEED_STATUS -d DBCONFIG -n SLACK_CONFIG -a ASANA_CONFIG -i ASANA_PROJECT_ID -f INPUT_LIST

Parameters

-h, --help : show this help message and exit

-t, --table_name TABLE_NAME : Table name for igf id lookup

-p, --pipeline_name PIPELINE_NAME : Pipeline name for seed modification

-s, --seed_status SEED_STATUS : New seed status for pipeline_seed table

-d, --dbconfig DBCONFIG : Database configuration file path

-n, --slack_config SLACK_CONFIG : Slack configuration file path

-a, --asana_config ASANA_CONFIG : Asana configuration file path

-i, --asana_project_id ASANA_PROJECT_ID : Asana project id

-f, --input_list INPUT_LIST : IGF id list file

1.3 Samplesheet processing

1.3.1 Divide samplesheet data

Usage

divide_samplesheet.py
[-h] -i SAMPLESHEET_FILE -d OUTPUT_DIR [-p]

Parameters

-h, --help : show this help message and exit

-i, -samplesheet_file SAMPLESHEET_FILE : Illumina format samplesheet file -d, –output_dir OUTPUT_DIR : Output directory for writing samplesheet file -p, –print_stats : Print available stats for the samplesheet and exit

1.3.2 Reformat samplesheet for demultiplexing

Usage

reformatSampleSheet.py
[-h] -i SAMPLESHEET_FILE -f RUNINFOXML_FILE [-r] -o OUTPUT_FILE

Parameters

-h, --help : show this help message and exit

-i, --samplesheet_file SAMPLESHEET_FILE : Illumina format samplesheet file

-f, --runinfoxml_file RUNINFOXML_FILE : Illumina RunInfo.xml file

-r, --revcomp_index : Reverse complement HiSeq and NextSeq index2 column, default: True

-o, --output_file OUTPUT_FILE : Reformatted samplesheet file

1.3.3 Calculate basesmask for demultiplexing

Usage

makeBasesMask.py
[-h] -s SAMPLESHEET_FILE -r RUNINFO_FILE [-a READ_OFFSET] [-b INDEX_OFFSET]

Parameters

-h, --help : show this help message and exit

-s, --samplesheet_file SAMPLESHEET_FILE : Illumina format samplesheet file

-r, --runinfo_file RUNINFO_FILE : Illumina format RunInfo.xml file

-a, --read_offset READ_OFFSET : Extra sequencing cycle for reads, default: 1

-b, --index_offset INDEX_OFFSET : Extra sequencing cycle for index, default: 0

1.4 Create or modify data to database

1.4.1 Clean up data from existing database and create new tables

Usage

clean_and_rebuild_database.py
[-h] -d DBCONFIG_PATH -s SLACK_CONFIG

Parameters

-h, --help	: Show this help message and exit
-d, --dbconfig_path	: Database configuration json file
-s, --slack_config	: Slack configuration json file

1.4.2 Load flowcell runs to database

Usage

load_flowcell_rules_data.py
[-h] -f FLOWCELL_DATA [-u] -d DBCONFIG_PATH -s SLACK_CONFIG

Parameters

-h, --help	: Show this help message and exit
-f, --flowcell_data	: Flowcell rules data json file
-u, --update	: Update existing flowcell rules data, default: False
-d, --dbconfig_path	: Database configuration json file
-s, --slack_config	: Slack configuration json file

1.4.3 Load pipeline configuration to database

Usage

load_pipeline_data.py
[-h] -p PIPELINE_DATA [-u] -d DBCONFIG_PATH -s SLACK_CONFIG

Paramaters

-h, --help	: Show this help message and exit
-p, --pipeline_data	: Pipeline data json file
-u, --update	: Update existing platform data, default: False
-d, --dbconfig_path	: Database configuration json file
-s, --slack_config	: Slack configuration json file

1.4.4 Load sequencing platform information to database

Usage

load_platform_data.py [-h] -p PLATFORM_DATA [-u] -d DBCONFIG_PATH -s SLACK_CONFIG

Parameters

-h, --help	: Show this help message and exit
-p, --platform_data	: Platform data json file
-u, --update	: Update existing platform data, default: False
-d, --dbconfig_path	: Database configuration json file
-s, --slack_config	: Slack configuration json file

1.4.5 Load sequencing run information to database from a text input

Usage

load_seqrun_data.py [-h] -p SEQRUN_DATA -d DBCONFIG_PATH -s SLACK_CONFIG

Parameters

-h, --help	: Show this help message and exit
-p, --seqrun_data	: Seqrun data json file
-d, --dbconfig_path	: Database configuration json file
-s, --slack_config	: Slack configuration json file

1.4.6 Load file entries and build collection in database

Usage

load_files_collecion_to_db.py
[-h] -f COLLECTION_FILE_DATA -d DBCONFIG_PATH [-s]

Parameters

-h, --help	: show this help message and exit
-f, --collection_file_data COLLECTION_FILE_DATA	: Collection file data json file
-d, --dbconfig_path DBCONFIG_PATH	: Database configuration json file
-s, --calculate_checksum	: Toggle file checksum calculation

1.5 Check Storage utilisation

1.5.1 Calculate disk usage summary

Usage

calculate_disk_usage_summary.py
[-h] -p DISK_PATH [-c] [-r REMOTE_SERVER] -o OUTPUT_PATH

Parameters

-h, --help	: show this help message and exit
-p, --disk_path DISK_PATH	: List of disk path for summary calculation

-c, --copy_to_remoter : Toggle file copy to remote server

-r, --remote_server REMOTE_SERVER : Remote server address

-o, --output_path OUTPUT_PATH : Output directory path

1.5.2 Calculate disk usage for a top level directory

Usage

calculate_sub_directory_usage.py
[-h] -p DIRECTORY_PATH [-c] [-r REMOTE_SERVER] -o OUTPUT_FILEPATH

Parameters

-h, --help : show this help message and exit

-p, --directory_path DIRECTORY_PATH : A directory path for sub directory lookup

-c, --copy_to_remoter : Toggle file copy to remote server

-r, --remote_server REMOTE_SERVER : Remote server address

-o, --output_filepath OUTPUT_FILEPATH : Output gviz file path

1.5.3 Merge disk usage summary file and build a gviz json

Usage

merge_disk_usage_summary.py
[-h] -f CONFIG_FILE [-l LABEL_FILE] [-c] [-r REMOTE_SERVER] -o OUTPUT_FILEPATH

Parameters

-h, --help : show this help message and exit

-f, --config_file CONFIG_FILE : A configuration json file for disk usage summary

-l, --label_file LABEL_FILE : A json file for disk label name

-c, --copy_to_remoter : Toggle file copy to remote server

-r, --remote_server REMOTE_SERVER : Remote server address

-o, --output_filepath OUTPUT_FILEPATH : Output gviz file path

1.5.4 Seed analysis pipeline

A script for finding new experiment entries for seeding analysis pipeline

Usage

find_and_seed_new_analysis.py
[-h] -d DBCONFIG_PATH -s SLACK_CONFIG -p PIPELINE_NAME -t FASTQ_TYPE -f PROJECT_NAME_FILE [-m SPECIES_NAME] [-l LIBRARY_SOURCE]

Parameters

-h, --help : show this help message and exit

-d , –dbconfig_path DBCONFIG_PATH : Database configuration json file -s , –slack_config SLACK_CONFIG : Slack configuration json file -p , –pipeline_name PIPELINE_NAME : IGF pipeline name -t , –fastq_type FASTQ_TYPE : Fastq collection type -f , –project_name_file PROJECT_NAME_FILE : File containing project names for seeding analysis pipeline -m ,

–species_name SPECIES_NAME : Species name to filter analysis -l , –library_source LIBRARY_SOURCE : Library source to filter analysis

CHAPTER

TWO

LIST OF PYTHON CLASSES AND FUNCTIONS

2.1 IGF database schema and api

2.1.1 Database schema

class igf_data.igfdb.igfTables.**Analysis**(***kwargs*)

A table for loading analysis design information

Parameters

- **analysis_id** – An integer id for analysis table
- **project_id** – A required integer id from project table (foreign key)
- **analysis_type** – An optional string field of 120chrs to specify analysis type
- **analysis_description** – An optional json description for analysis

class igf_data.igfdb.igfTables.**Collection**(***kwargs*)

A table for loading collection information

Parameters

- **collection_id** – An integer id for collection table
- **name** – A required string to specify collection name, allowed length 70
- **type** – A required string to specify collection type, allowed length 50
- **table** – An optional enum list to specify collection table information, default unknown, allowed values are sample, experiment, run, file, project, seqrun and unknown
- **date_stamp** – An optional timestamp column to record entry creation or modification time, default current timestamp

class igf_data.igfdb.igfTables.**Collection_attribute**(***kwargs*)

A table for loading collection attributes

Parameters

- **collection_attribute_id** – An integer id for collection_attribute table
- **attribute_name** – An optional string attribute name, allowed length 200
- **attribute_value** – An optional string attribute value, allowed length 200
- **collection_id** – An integer id from collection table (foreign key)

class igf_data.igfdb.igfTables.**Collection_group**(***kwargs*)

A table for linking files to the collection entries

Parameters

- **collection_group_id** – An integer id for collection_group table

- `collection_id` – A required integer id from collection table (foreign key)
- `file_id` – A required integer id from file table (foreign key)

class igf_data.igfdb.igfTables.**Experiment**(***kwargs*)

A table for loading experiment (unique combination of sample, library and platform) information.

Parameters

- `experiment_id` – An integer id for experiment table
- `experiment_igf_id` – A required string as experiment id specific to IGF team, allowed length 40
- `project_id` – A required integer id from project table (foreign key)
- `sample_id` – A required integer id from sample table (foreign key)
- `library_name` – A required string to specify library name, allowed length 50
- `library_source` – An optional enum list to specify library source information, default is UNKNOWN, allowed values are
  - GENOMIC
  - TRANSCRIPTOMIC
  - GENOMIC_SINGLE_CELL
  - TRANSCRIPTOMIC_SINGLE_CELL
  - METAGENOMIC
  - METATRANSCRIPTOMIC
  - SYNTHETIC
  - VIRAL_RNA
  - UNKNOWN
- `library_strategy` – An optional enum list to specify library strategy information, default is UNKNOWN, allowed values are
  - WGS
  - WXS
  - WGA
  - RNA-SEQ
  - CHIP-SEQ
  - ATAC-SEQ
  - MIRNA-SEQ
  - NCRNA-SEQ
  - FL-CDNA
  - EST
  - HI-C
  - DNASE-SEQ
  - WCS
  - RAD-SEQ
  - CLONE
  - POOLCLONE

  - AMPLICON
  - CLONEEND
  - FINISHING
  - MNASE-SEQ
  - DNASE-HYPERSENSITIVITY
  - BISULFITE-SEQ
  - CTS
  - MRE-SEQ
  - MEDIP-SEQ
  - MBD-SEQ
  - TN-SEQ
  - VALIDATION
  - FAIRE-SEQ
  - SELEX
  - RIP-SEQ
  - CHIA-PET
  - SYNTHETIC-LONG-READ
  - TARGETED-CAPTURE
  - TETHERED
  - NOME-SEQ
  - CHIRP SEQ
  - 4-C-SEQ
  - 5-C-SEQ
  - UNKNOWN

- **experiment_type** – An optional enum list as experiment type information, default is UNKNOWN, allowed values are
  - POLYA-RNA
  - POLYA-RNA-3P
  - TOTAL-RNA
  - SMALL-RNA
  - WGS
  - WGA
  - WXS
  - WXS-UTR
  - RIBOSOME-PROFILING
  - RIBODEPLETION
  - 16S
  - NCRNA-SEQ
  - FL-CDNA

- EST
- HI-C
- DNASE-SEQ
- WCS
- RAD-SEQ
- CLONE
- POOLCLONE
- AMPLICON
- CLONEEND
- FINISHING
- DNASE-HYPERSENSITIVITY
- RRBS-SEQ
- WGBS
- CTS
- MRE-SEQ
- MEDIP-SEQ
- MBD-SEQ
- TN-SEQ
- VALIDATION
- FAIRE-SEQ
- SELEX
- RIP-SEQ
- CHIA-PET
- SYNTHETIC-LONG-READ
- TARGETED-CAPTURE
- TETHERED
- NOME-SEQ
- CHIRP-SEQ
- 4-C-SEQ
- 5-C-SEQ
- METAGENOMIC
- METATRANSCRIPTOMIC
- TF
- H3K27ME3
- H3K27AC
- H3K9ME3
- H3K36ME3
- H3F3A
- H3K4ME1

  - H3K79ME2
  - H3K79ME3
  - H3K9ME1
  - H3K9ME2
  - H4K20ME1
  - H2AFZ
  - H3AC
  - H3K4ME2
  - H3K4ME3
  - H3K9AC
  - HISTONE-NARROW
  - HISTONE-BROAD
  - CHIP-INPUT
  - ATAC-SEQ
  - TENX-TRANSCRIPTOME-3P
  - TENX-TRANSCRIPTOME-5P
  - DROP-SEQ-TRANSCRIPTOME
  - UNKNOWN
- **library_layout** – An optional enum list to specify library layout, default is UNONWN, allowed values are
  - SINGLE
  - PAIRED
  - UNKNOWN
- **status** – An optional enum list to specify experiment status, default is ACTIVE, allowed values are
  - ACTIVE
  - FAILED
  - WITHDRAWN
- **date_created** – An optional timestamp column to record entry creation or modification time, default current timestamp
- **platform_name** – An optional enum list to specify platform model, default is UNKNOWN, allowed values are
  - HISEQ250
  - HISEQ4000
  - MISEQ
  - NEXTSEQ
  - NOVASEQ6000
  - NANOPORE_MINION
  - DNBSEQ-G400
  - DNBSEQ-G50

  - DNBSEQ-T7
  - NEXTSEQ2000
  - SEQUEL2
  - UNKNOWN

`class igf_data.igfdb.igfTables.`**`Experiment_attribute`**`(**kwargs)`

A table for loading experiment attributes

Parameters

- **`experiment_attribute_id`** – An integer id for experiment_attribute table
- **`attribute_name`** – An optional string attribute name, allowed length 30
- **`attribute_value`** – An optional string attribute value, allowed length 50
- **`experiment_id`** – An integer id from experiment table (foreign key)

`class igf_data.igfdb.igfTables.`**`File`**`(**kwargs)`

A table for loading file information

Parameters

- **`file_id`** – An integer id for file table
- **`file_path`** – A required string to specify file path information, allowed length 500
- **`location`** – An optional enum list to specify storage location, default UNKNOWN, allowed values are
  - ORWELL
  - HPC_PROJECT
  - ELIOT
  - IRODS
  - UNKNOWN
- **`status`** – An optional enum list to specify experiment status, default is ACTIVE, allowed values are
  - ACTIVE
  - FAILED
  - WITHDRAWN
- **`md5`** – An optional string to specify file md5 value, allowed length 33
- **`size`** – An optional string to specify file size, allowed value 15
- **`date_created`** – An optional timestamp column to record file creation time, default current timestamp
- **`date_updated`** – An optional timestamp column to record file modification time, default current timestamp

`class igf_data.igfdb.igfTables.`**`File_attribute`**`(**kwargs)`

A table for loading file attributes

Parameters

- **`file_attribute_id`** – An integer id for file_attribute table
- **`attribute_name`** – An optional string attribute name, allowed length 30
- **`attribute_value`** – An optional string attribute value, allowed length 50
- **`file_id`** – An integer id from file table (foreign key)

class igf_data.igfdb.igfTables.**Flowcell_barcode_rule**(***kwargs*)

A table for loading flowcell specific barcode rules information

Parameters

- **flowcell_rule_id** – An integer id for flowcell_barcode_rule table
- **platform_id** – An integer id for platform table (foreign key)
- **flowcell_type** – A required string as flowcell type name, allowed length 50
- **index_1** – An optional enum list as index_1 specific rule, default UNKNOWN, allowed values are
  - NO_CHANGE
  - REVCOMP
  - UNKNOWN
- **index_2** – An optional enum list as index_2 specific rule, default UNKNOWN, allowed values are
  - NO_CHANGE
  - REVCOMP
  - UNKNOWN

class igf_data.igfdb.igfTables.**History**(***kwargs*)

A table for loading history information

Parameters

- **log_id** – An integer id for history table
- **log_type** – A required enum value to specify log type, allowed values are
  - CREATED
  - MODIFIED
  - DELETED
- **table_name** – A required enum value to specify table information, allowed values are
  - PROJECT
  - USER
  - SAMPLE
  - EXPERIMENT
  - RUN
  - COLLECTION
  - FILE
  - PLATFORM
  - PROJECT_ATTRIBUTE
  - EXPERIMENT_ATTRIBUTE
  - COLLECTION_ATTRIBUTE
  - SAMPLE_ATTRIBUTE
  - RUN_ATTRIBUTE
  - FILE_ATTRIBUTE

- `log_date` – An optional timestamp column to record file creation or modification time, default current timestamp
- `message` – An optional text field to specify message

`class igf_data.igfdb.igfTables.Pipeline(**kwargs)`

A table for loading pipeline information

Parameters

- `pipeline_id` – An integer id for pipeline table
- `pipeline_name` – A required string to specify pipeline name, allowed length 50
- `pipeline_db` – A required string to specify pipeline database url, allowed length 200
- `pipeline_init_conf` – An optional json field to specify initial pipeline configuration
- `pipeline_run_conf` – An optional json field to specify modified pipeline configuration
- `pipeline_type` – An optional enum list to specify pipeline type, default EHIVE, allowed values are
  - EHIVE
  - UNKNOWN
  - AIRFLOW
  - NEXTFLOW
- `is_active` – An optional enum list to specify the status of pipeline, default Y allowed values are Y and N
- `date_stamp` – An optional timestamp column to record file creation or modification time, default current timestamp

`class igf_data.igfdb.igfTables.Pipeline_seed(**kwargs)`

A table for loading pipeline seed information

Parameters

- `pipeline_seed_id` – An integer id for pipeline_seed table
- `seed_id` – A required integer id
- `seed_table` – An optional enum list to specify seed table information, default unknown, allowed values project, sample, experiment, run, file, seqrun, collection and unknown
- `pipeline_id` – An integer id from pipeline table (foreign key)
- `status` –

  An optional enum list to specify the status of pipeline, default UNKNOWN,
  allowed values are

  - SEEDED
  - RUNNING
  - FINISHED
  - FAILED
  - UNKNOWN
- `date_stamp` – An optional timestamp column to record file creation or modification time, default current timestamp

class `igf_data.igfdb.igfTables.`**`Platform`**(***kwargs*)

A table for loading sequencing platform information

Parameters

- **platform_id** – An integer id for platform table
- **platform_igf_id** – A required string as platform id specific to IGF team, allowed length 10
- **model_name** – A required enum list to specify platform model, allowed values are
  - HISEQ2500
  - HISEQ4000
  - MISEQ
  - NEXTSEQ
  - NOVASEQ6000
  - NANOPORE_MINION
  - DNBSEQ-G400
  - DNBSEQ-G50
  - DNBSEQ-T7
- **vendor_name** – A required enum list to specify vendor's name, allowed values are
  - ILLUMINA
  - NANOPORE
  - MGI
- **software_name** – A required enum list for specifying platform software, allowed values are
  - RTA
  - UNKNOWN
- **software_version** – A optional software version number, default is UNKNOWN
- **date_created** – An optional timestamp column to record entry creation time, default current timestamp

class `igf_data.igfdb.igfTables.`**`Project`**(***kwargs*)

A table for loading project information

Parameters

- **project_id** – An integer id for project table
- **project_igf_id** – A required string as project id specific to IGF team, allowed length 50
- **project_name** – An optional string as project name
- **start_timestamp** – An optional timestamp for project creation, default current timestamp
- **description** – An optional text column to document project description
- **deliverable** – An enum list to document project deliverable, default FASTQ,allowed entries are
  - FASTQ
  - ALIGNMENT

    - ANALYSIS

- **status** – An enum list for project status, default ACTIVE, allowed entries are
    - ACTIVE
    - FINISHED
    - WITHDRAWN

`class igf_data.igfdb.igfTables.`**`ProjectUser`**`(**kwargs)`

A table for linking users to the projects

Parameters

- **project_user_id** – An integer id for project_user table
- **project_id** – An integer id for project table (foreign key)
- **user_id** – An integer id for user table (foreign key)
- **data_authority** – An optional enum value to denote primary user for the project, allowed value T

`class igf_data.igfdb.igfTables.`**`Project_attribute`**`(**kwargs)`

A table for loading project attributes

Parameters

- **project_attribute_id** – An integer id for project_attribute table
- **attribute_name** – An optional string attribute name, allowed length 50
- **attribute_value** – An optional string attribute value, allowed length 50
- **project_id** – An integer id from project table (foreign key)

`class igf_data.igfdb.igfTables.`**`Run`**`(**kwargs)`

A table for loading run (unique combination of experiment, sequencing flowcell and lane) information

Parameters

- **run_id** – An integer id for run table
- **run_igf_id** – A required string as run id specific to IGF team, allowed length 70
- **experiment_id** – A required integer id from experiment table (foreign key)
- **seqrun_id** – A required integer id from seqrun table (foreign key)
- **status** – An optional enum list to specify experiment status, default is ACTIVE, allowed values are
    - ACTIVE
    - FAILED
    - WITHDRAWN
- **lane_number** – A required enum list for specifying lane information, allowed values 1, 2, 3, 4, 5, 6, 7 and 8
- **date_created** – An optional timestamp column to record entry creation time, default current timestamp

`class igf_data.igfdb.igfTables.`**`Run_attribute`**`(**kwargs)`

A table for loading run attributes

Parameters

- **run_attribute_id** – An integer id for run_attribute table
- **attribute_name** – An optional string attribute name, allowed length 30

- `attribute_value` – An optional string attribute value, allowed length 50
- `run_id` – An integer id from run table (foreign key)

`class igf_data.igfdb.igfTables.Sample(**kwargs)`

A table for loading sample information

Parameters

- `sample_id` – An integer id for sample table
- `sample_igf_id` – A required string as sample id specific to IGF team, allowed length 20
- `sample_submitter_id` – An optional string as sample name from user, allowed value 40
- `taxon_id` – An optional integer NCBI taxonomy information for sample
- `scientific_name` – An optional string as scientific name of the species
- `species_name` – An optional string as the species name (genome build code) information
- `donor_anonymized_id` – An optional string as anonymous donor name
- `description` – An optional string as sample description
- `phenotype` – An optional string as sample phenotype information
- `sex` – An optional enum list to specify sample sex, default UNKNOWN, allowed values are
  - FEMALE
  - MALE
  - MIXED
  - UNKNOWN
- `status` – An optional enum list to specify sample status, default ACTIVE, allowed values are
  - ACTIVE
  - FAILED
  - WITHDRAWS
- `biomaterial_type` – An optional enum list as sample biomaterial type, default UNKNOWN, allowed values are
  - PRIMARY_TISSUE
  - PRIMARY_CELL
  - PRIMARY_CELL_CULTURE
  - CELL_LINE
  - SINGLE_NUCLEI
  - UNKNOWN
- `cell_type` – An optional string to specify sample cell_type information, if biomaterial_type is PRIMARY_CELL or PRIMARY_CELL_CULTURE
- `tissue_type` – An optional string to specify sample tissue information, if biomaterial_type is PRIMARY_TISSUE
- `cell_line` – An optional string to specify cell line information ,if biomaterial_type is CELL_LINE

- `date_created` – An optional timestamp column to specify entry creation date, default current timestamp
- `project_id` – An integer id for project table (foreign key)

class igf_data.igfdb.igfTables.**Sample_attribute**(***kwargs*)

A table for loading sample attributes

Parameters

- `sample_attribute_id` – An integer id for sample_attribute table
- `attribute_name` – An optional string attribute name, allowed length 50
- `attribute_value` – An optional string attribute value, allowed length 50
- `sample_id` – An integer id from sample table (foreign key)

class igf_data.igfdb.igfTables.**Seqrun**(***kwargs*)

A table for loading sequencing run information

Parameters

- `seqrun_id` – An integer id for seqrun table
- `seqrun_igf_id` – A required string as seqrun id specific to IGF team, allowed length 50
- `reject_run` – An optional enum list to specify rejected run information ,default N, allowed values Y and N
- `date_created` – An optional timestamp column to record entry creation time, default current timestamp
- `flowcell_id` – A required string column for storing flowcell_id information, allowed length 20
- `platform_id` – An integer platform id (foreign key)

class igf_data.igfdb.igfTables.**Seqrun_attribute**(***kwargs*)

A table for loading seqrun attributes

Parameters

- `seqrun_attribute_id` – An integer id for seqrun_attribute table
- `attribute_name` – An optional string attribute name, allowed length 50
- `attribute_value` – An optional string attribute value, allowed length 100
- `seqrun_id` – An integer id from seqrun table (foreign key)

class igf_data.igfdb.igfTables.**Seqrun_stats**(***kwargs*)

A table for loading sequencing stats information

Parameters

- `seqrun_stats_id` – An integer id for seqrun_stats table
- `seqrun_id` – An integer seqrun id (foreign key)
- `lane_number` – A required enum list for specifying lane information, allowed values are 1, 2, 3, 4, 5, 6, 7 and 8
- `bases_mask` – An optional string field for storing bases mask information
- `undetermined_barcodes` – An optional json field to store barcode info for undetermined samples
- `known_barcodes` – An optional json field to store barcode info for known samples

- **undetermined_fastqc** – An optional json field to store qc info for undetermined samples

class igf_data.igfdb.igfTables.**User**(***kwargs*)

A table for loading user information

Parameters

- **user_id** – An integer id for user table
- **user_igf_id** – An optional string as user id specific to IGF team, allowed length 10
- **name** – A required string as user name, allowed length 30
- **email_id** – A required string as email id, allowed length 40
- **username** – A required string as IGF username, allowed length 20
- **hpc_username** – An optional string as Imperial College's HPC login name, allowed length 20
- **twitter_user** – An optional string as twitter user name, allowed length 20
- **category** – An optional enum list as user category, default NON_HPC_USER, allowed values are
  - HPC_USER
  - NON_HPC_USER
  - EXTERNAL
- **status** – An optional enum list as user status, default is ACTIVE, allowed values are
  - ACTIVE
  - BLOCKED
  - WITHDRAWN
- **date_created** – An optional timestamp, default current timestamp
- **password** – An optional string field to store encrypted password
- **encryption_salt** – An optional string field to store encryption salt
- **ht_password** – An optional field to store password for htaccess

2.1.2 Database adaptor api

Base adaptor

class igf_data.igfdb.baseadaptor.**BaseAdaptor**(***data*)

The base adaptor class

divide_data_to_table_and_attribute(*data*, *required_column*, *table_columns*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for main and attribute tables

Parameters

- **data** – a dictionary or dataframe containing the data
- **required_column** – column to add to the attribute table, it must be part of the data
- **table_columns** – required columns for the main table
- **attribute_name_column** – column label for attribute name

- **attribute_value_column** – column label for attribute value

Returns
Two pandas dataframes, one for main table and one for attribute tables

`fetch_records`(*query*, *output_mode='dataframe'*)
A method for fetching records using a query

Parameters

- **query** – A sqlalchmeny query object
- **output_mode** – dataframe / object / one / one_or_none

Returns
A pandas dataframe for dataframe mode and a generator object for object mode

`fetch_records_by_column`(*table*, *column_name*, *column_id*, *output_mode*)
A method for fetching record with the column

Parameters

- **table** – table name
- **column_name** – a column name
- **column_id** – a column id value
- **output_mode** – dataframe / object / one / one_or_none

`fetch_records_by_multiple_column`(*table*, *column_data*, *output_mode*)
A method for fetching record with the column

Parameters

- **table** – table name
- **column_dict** – a dictionary of column_names: column_value
- **output_mode** – dataframe / object/ one / one_or_none

`get_attributes_by_dbid`(*attribute_table*, *linked_table*, *linked_column_name*, *db_id*)
A method for fetching attribute records for a specific attribute table with a db_id linked as foreign key

Parameters

- **attribute_table** – A attribute table object
- **linked_table** – A main table object
- **linked_column_name** – A table name to link main table
- **db_id** – A unique id to link main table

:returns a dataframe of records

`get_table_columns`(*table_name*, *excluded_columns*)
A method for fetching the columns for table table_name

Parameters

- **table_name** – a table class name
- **excluded_columns** – a list of column names to exclude from output

`map_foreign_table_and_store_attribute`(*data*, *lookup_table*, *lookup_column_name*, *target_column_name*)
A method for mapping foreign key id to the new column

Parameters

- **data** – a data dictionary or pandas series, to be stored in attribute table

- **lookup_table** – a table class to look for the foreign key id
- **lookup_column_name** – a string or a list of column names which will be used to link the data frame with lookup_table, this column will be removed from the output series
- **target_column_name** – column name for the foreign key id

Returns
A data series

`store_attributes`(*attribute_table*, *data*, *linked_column=''*, *db_id=''*, *mode='serial'*)
A method for storing attributes

Parameters

- **attribute_table** – a attribute table name
- **linked_column** – a column name to link the db_id to attribute table
- **db_id** – a db_id to link the attribute records
- **mode** – serial / bulk

`store_records`(*table*, *data*, *mode='serial'*)
A method for loading data to table

Parameters
table – name of the table class

:param data : pandas dataframe or a list of dictionary :param mode : serial / bulk

Project adaptor

class `igf_data.igfdb.projectadaptor.ProjectAdaptor`(***data*)
An adaptor class for Project, ProjectUser and Project_attribute tables

`assign_user_to_project`(*data*, *required_project_column='project_igf_id'*, *required_user_column='email_id'*, *data_authority_column='data_authority'*, *autosave=True*)
Load data to ProjectUser table

Parameters

- **data** – A list of dictionaries, each containing 'project_igf_id' and 'user_igf_id' as key with relevent igf ids as the values. An optional key 'data_authority' with boolean value can be provided to set the user as the data authority of the project E.g. [{'project_igf_id': val, 'email_id': val, 'data_authority':True},]
- **required_project_column** – Name of the project id column, default project_igf_id
- **required_user_column** – Name of the user id column, default email_id
- **data_authority_column** – Name of the data_authority column, default data_authority
- **autosave** – A toggle for autocommit to db, default True

Returns
None

`check_data_authority_for_project`(*project_igf_id*)
A method for checking user data authority for existing projects

Parameters
project_igf_id – An unique project igf id

Returns
True if data authority exists for project or false

`check_existing_project_user`(*project_igf_id*, *email_id*)

A method for checking existing project use info in database

Parameters

- `project_igf_id` – A project_igf_id
- `email_id` – An email_id

Returns
True if the file is present in db or False if its not

`check_project_attributes`(*project_igf_id*, *attribute_name*)

A method for checking existing project attribute in database

Parameters

- `project_igf_id` – An unique project igf id
- `attribute_name` – An attribute name

:return A boolean value

`check_project_records_igf_id`(*project_igf_id*, *target_column_name='project_igf_id'*)

A method for checking existing data for Project table

Parameters

- `project_igf_id` – Project igf id name
- `target_column_name` – Name of the project id column, default project_igf_id

Returns
True if the file is present in db or False if its not

`count_project_samples`(*project_igf_id*, *only_active=True*)

A method for counting total number of samples for a project

Parameters

- `project_igf_id` – A project id
- `only_active` – Toggle for including only active projects, default is True

Returns
A int sample count

`divide_data_to_table_and_attribute`(*data*, *required_column='project_igf_id'*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for Project and Project_attribute tables

Parameters

- `data` – A list of dictionaries or a pandas dataframe
- `required_column` – Name of the required column, default project_igf_id
- `attribute_name_column` – Value for attribute name column, default attribute_name
- `attribute_value_column` – Valye for attribute value column, default attribute_value

Returns
A project dataframe and a project attribute dataframe

`fetch_all_project_igf_ids`(*output_mode='dataframe'*)

A method for fetching a list of all project igf ids

Parameters
`output_mode` – Output mode, default dataframe

`fetch_data_authority_for_project`(*project_igf_id*)

A method for fetching user data authority for existing projects

Parameters
`project_igf_id` – An unique project igf id

Returns
A user object or None, if no entry found

`fetch_project_records_igf_id`(*project_igf_id*, *target_column_name='project_igf_id'*)

A method for fetching data for Project table

Parameters

- `project_igf_id` – an igf id
- `output_mode` – dataframe / object / one

Returns
Records from project table

`fetch_project_samples`(*project_igf_id*, *only_active=True*, *output_mode='object'*)

A method for fetching all the samples for a specific project

Parameters

- `project_igf_id` – A project id
- `only_active` – Toggle for including only active projects, default is True
- `output_mode` – Output mode, default object

Returns
Depends on the output_mode, a generator expression, dataframe or an object

`get_project_attributes`(*project_igf_id*, *linked_column_name='project_id'*, *attribute_name=''*)

A method for fetching entries from project attribute table

Parameters

- `project_igf_id` – A project_igf_id string
- `attribute_name` – An attribute name, default in None
- `linked_column_name` – A column name for linking attribute table

:returns dataframe of records

`get_project_user_info`(*output_mode='dataframe'*, *project_igf_id=''*)

A method for fetching information from Project, User and ProjectUser table

Parameters
`project_igf_id` – a project igf id

:param output_mode : dataframe / object :returns: Records for project user

`store_project_and_attribute_data`(*data*, *autosave=True*)

A method for dividing and storing data to project and attribute_table

Parameters

- `data` – A list of data or a pandas dataframe
- `autosave` – A toggle for autocommit, default True

Returns
None

store_project_attributes(*data*, *project_id=''*, *autosave=False*)

A method for storing data to Project_attribute table

Parameters

- **data** – A pandas dataframe
- **project_id** – Project id for attribute table, default ‘’
- **autosave** – A toggle for autocommit, default False

Returns
None

store_project_data(*data*, *autosave=False*)

Load data to Project table

Parameters

- **data** – A list of data or a pandas dataframe
- **autosave** – A toggle for autocommit, default False

Returns
None

User adaptor

class igf_data.igfdb.useradaptor.**UserAdaptor**(***data*)

An adaptor class for table User

check_user_records_email_id(*email_id*)

A method for checking existing user data in db

Parameters
email_id – An email id

Returns
True if the file is present in db or False if its not

fetch_user_records_email_id(*user_email_id*)

A method for fetching data for User table

Parameters
user_email_id – an email id

Returns
user object

fetch_user_records_igf_id(*user_igf_id*)

A method for fetching data for User table

Parameters
user_igf_id – an igf id

Returns
user object

store_user_data(*data*, *autosave=True*)

Load data to user table

Parameters

- **data** – A pandas dataframe
- **autosave** – A toggle for autocommit, default True

Returns
None

Sample adaptor

class igf_data.igfdb.sampleadaptor.**SampleAdaptor**(***data*)

An adaptor class for Sample and Sample_attribute tables

check_project_and_sample(*project_igf_id*, *sample_igf_id*)

A method for checking existing project and sample igf id combination in sample table

Parameters

- **project_igf_id** – A project igf id string
- **sample_igf_id** – A sample igf id string

Returns
True if target entry is present or return False

check_sample_records_igf_id(*sample_igf_id*, *target_column_name='sample_igf_id'*)

A method for checking existing data for sample table

Parameters

- **sample_igf_id** – an igf id
- **target_column_name** – name of the target lookup column, default sample_igf_id

Returns
True if the file is present in db or False if its not

divide_data_to_table_and_attribute(*data*, *required_column='sample_igf_id'*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for Sample and Sample_attribute tables

Parameters

- **data** – A list of dictionaries or a pandas dataframe
- **table_columns** – List of table column names, default None
- **required_column** – column name to add to the attribute data
- **attribute_name_column** – label for attribute name column
- **attribute_value_column** – label for attribute value column

Returns
Two pandas dataframes, one for Sample and another for Sample_attribute table

fetch_sample_project(*sample_igf_id*)

A method for fetching project information for the sample

Parameters
sample_igf_id – A sample_igf_id for database lookup

Returns
A project_igf_id or None, if not found

fetch_sample_records_igf_id(*sample_igf_id*, *target_column_name='sample_igf_id'*)

A method for fetching data for Sample table

Parameters

- **sample_igf_id** – A sample igf id

- **output_mode** – dataframe, object, one or on_on_none

Returns
An object or dataframe, based on the output_mode

`fetch_seqrun_and_platform_list_for_sample_id`(*sample_igf_id*, *output_mode='dataframe'*)
A method for fetching seqrn and platform information for a sample

Parameters

- **sample_igf_id** – Sample igf id
- **output_mode** – Output format, default 'dataframe'

Returns
A object or dataframe

`store_sample_and_attribute_data`(*data*, *autosave=True*)
A method for dividing and storing data to sample and attribute table

`store_sample_attributes`(*data*, *sample_id=''*, *autosave=False*)
A method for storing data to Sample_attribute table

Parameters

- **data** – A dataframe or list of dictionary containing the Sample_attribute data
- **sample_id** – An optional parameter to link the sample attributes to a specific sample

Returns
None

`store_sample_data`(*data*, *autosave=False*)
Load data to Sample table

Parameters

- **data** – A dataframe or list of dictionary containing the data
- **autosave** – A toggle for autocommit, default False

Returns
None

Experiment adaptor

class igf_data.igfdb.experimentadaptor.**ExperimentAdaptor**(***data*)
An adaptor class for Experiment and Experiment_attribute tables

`check_experiment_records_id`(*experiment_igf_id*, *target_column_name='experiment_igf_id'*)
A method for checking existing data for Experiment table

Parameters

- **experiment_igf_id** – an igf id
- **target_column_name** – a column name, default experiment_igf_id

Returns
True if the file is present in db or False if its not

`divide_data_to_table_and_attribute`(*data*, *required_column='experiment_igf_id'*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)
A method for separating data for Experiment and Experiment_attribute tables

Parameters

- **data** – A list of dictionaries or a Pandas DataFrame
- **table_columns** – List of table column names, default None
- **required_column** – column name to add to the attribute data
- **attribute_name_column** – label for attribute name column
- **attribute_value_column** – label for attribute value column

Returns
Two pandas dataframes, one for Experiment and another for Experiment_attribute table

fetch_experiment_records_id(*experiment_igf_id*, *target_column_name='experiment_igf_id'*)
A method for fetching data for Experiment table

Parameters
- **experiment_igf_id** – an igf id
- **target_column_name** – a column name, default experiment_igf_id

Returns
Experiment object

fetch_project_and_sample_for_experiment(*experiment_igf_id*)
A method for fetching project and sample igf id information for an experiment

Parameters
experiment_igf_id – An experiment igf id string

Returns
Two strings, project igf id and sample igd id, or None if not found

fetch_runs_for_igf_id(*experiment_igf_id*, *include_active_runs=True*, *output_mode='dataframe'*)
A method for fetching all the runs for a specific experiment_igf_id

Parameters
- **experiment_igf_id** – An experiment_igf_id
- **include_active_runs** – Include only active runs, if its True, default True
- **output_mode** – Record fetch mode, default dataframe

Returns
Run records

fetch_sample_attribute_records_for_experiment_igf_id(*experiment_igf_id*, *output_mode='dataframe'*, *attribute_list=None*)
A method for fetching sample_attribute_records for a given experiment_igf_id

Parameters
- **experiment_igf_id** – An experiment_igf_id
- **output_mode** – Result output mode, default dataframe
- **attribute_list** – A list of attributes for database lookup, default None

Returns
an object or dataframe based on the output_mode

store_experiment_attributes(*data*, *experiment_id=''*, *autosave=False*)
A method for storing data to Experiment_attribute table

Parameters
- **data** – A list of dictionaries or a Pandas DataFrame for experiment attribute data

- **experiment_id** – An optional experiment_id to link attribute records
- **autosave** – A toggle for automatically saving data to db, default True

Returns
True

store_experiment_data(*data*, *autosave=False*)
Load data to Experiment table

Parameters
- **data** – A list of dictionaries or a Pandas DataFrame
- **autosave** – A toggle for automatically saving data to db, default True

Returns
None

store_project_and_attribute_data(*data*, *autosave=True*)
A method for dividing and storing data to experiment and attribute table

Parameters
- **data** – A list of dictionaries or a Pandas DataFrame
- **autosave** – A toggle for automatically saving data to db, default True

Returns
None

update_experiment_records_by_igf_id(*experiment_igf_id*, *update_data*, *autosave=True*)
A method for updating experiment records in database

Parameters
- **experiment_igf_id** – An igf ids for the experiment data lookup
- **update_data** – A dictionary containing the updated entries
- **autosave** – Toggle auto commit after database update, default True

Returns
None

Run adaptor

class igf_data.igfdb.runadaptor.**RunAdaptor**(***data*)
An adaptor class for Run and Run_attribute tables

check_run_records_igf_id(*run_igf_id*, *target_column_name='run_igf_id'*)
A method for existing data for Run table

Parameters
- **run_igf_id** – an igf id
- **target_column_name** – a column name, default run_igf_id

Returns
True if the file is present in db or False if its not

divide_data_to_table_and_attribute(*data*, *required_column='run_igf_id'*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)
A method for separating data for Run and Run_attribute tables

Parameters

- **data** – A list of dictionaries or a Pandas DataFrame
- **table_columns** – List of table column names, default None
- **required_column** – column name to add to the attribute data
- **attribute_name_column** – label for attribute name column
- **attribute_value_column** – label for attribute value column

Returns
Two pandas dataframes, one for Run and another for Run_attribute table

fetch_flowcell_and_lane_for_run(*run_igf_id*)
A run adapter method for fetching flowcell id and lane info for each run

Parameters
run_igf_id – A run igf id string

Returns
Flowcell id and lane number It will return None if no records found

fetch_project_sample_and_experiment_for_run(*run_igf_id*)
A method for fetching project, sample and experiment information for a run

Parameters
run_igf_id – A run igf id string

Returns
A list of three strings, or None if not found * project_igf_id * sample_igf_id * experiment_igf_id

fetch_run_records_igf_id(*run_igf_id*, *target_column_name='run_igf_id'*)
A method for fetching data for Run table

Parameters

- **run_igf_id** – an igf id
- **target_column_name** – a column name, default run_igf_id

Returns
Run record

fetch_sample_info_for_run(*run_igf_id*)
A method for fetching sample information linked to a run_igf_id

Parameters
run_igf_id – A run_igf_id to search database

Returns
Sample record

store_run_and_attribute_data(*data*, *autosave=True*)
A method for dividing and storing data to run and attribute table

Parameters

- **data** – A list of dictionaries or a Pandas DataFrame containing the run data
- **autosave** – A toggle for saving data automatically to db, default True

Returns
None

store_run_attributes(*data*, *run_id=''*, *autosave=False*)
A method for storing data to Run_attribute table

Parameters

- **data** – A list of dictionaries or a Pandas DataFrame containing the attribute data
- **autosave** – A toggle for saving data automatically to db, default True

Returns
None

`store_run_data`(*data*, *autosave=False*)

A method for loading data to Run table

Parameters

- **data** – A list of dictionaries or a Pandas DataFrame containing the attribute data
- **autosave** – A toggle for saving data automatically to db, default True

Returns
None

Collection adaptor

class igf_data.igfdb.collectionadaptor.`CollectionAdaptor`(***data*)

An adaptor class for Collection, Collection_group and Collection_attribute tables

`check_collection_attribute`(*collection_name*, *collection_type*, *attribute_name*)

A method for checking collection attribute records for an attribute_name

Parameters

- **collection_name** – A collection name
- **collection_type** – A collection type
- **attribute_name** – A collection attribute name

Returns
Boolean, True if record exists or False

`check_collection_records_name_and_type`(*collection_name*, *collection_type*)

A method for checking existing data for Collection table

Parameters

- **collection_name** – a collection name value
- **collection_type** – a collection type value

Returns
True if the file is present in db or False if its not

`create_collection_group`(*data*, *autosave=True*, *required_collection_column=('name', 'type')*, *required_file_column='file_path'*)

A function for creating collection group, a link between a file and a collection

Parameters

- **data** – A list dictionary or a Pandas DataFrame with following columns
  - name
  - type
  - file_path

  E.g. [{'name':'a collection name', 'type':'a collection type', 'file_path': 'path'},]
- **required_collection_column** – List of required column for fetching collection, default 'name','type'

- **required_file_column** – Required column for fetching file information, default file_path
- **autosave** – A toggle for saving changes to database, default True

Returns
None

create_or_update_collection_attributes(*data*, *autosave=True*)

A method for creating or updating collection attribute table, if the collection exists

Parameters

- **data** – A list of dictionaries, containing following entries * name * type * attribute_name * attribute_value
- **autosave** – A toggle for saving changes to database, default True

Returns
None

divide_data_to_table_and_attribute(*data*, *required_column=('name', 'type')*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for Collection and Collection_attribute tables

Parameters

- **data** – A list of dictionaries or a pandas dataframe
- **table_columns** – List of table column names, default None
- **required_column** – column name to add to the attribute data, default 'name', 'type'
- **attribute_name_column** – label for attribute name column, default attribute_name
- **attribute_value_column** – label for attribute value column, default attribute_value

Returns
Two pandas dataframes, one for Collection and another for Collection_attribute table

fetch_collection_name_and_table_from_file_path(*file_path*)

A method for fetching collection name and collection_table info using the file_path information. It will return None if the file doesn't have any collection present in the database

Parameters
file_path – A filepath info

Returns
Collection name and collection table for first collection group

fetch_collection_records_name_and_type(*collection_name*, *collection_type*, *target_column_name=('name', 'type')*)

A method for fetching data for Collection table

Parameters

- **collection_name** – a collection name value
- **collection_type** – a collection type value
- **target_column_name** – a list of columns, default is ['name','type']

Returns
Collection record

`get_collection_files`(*collection_name*, *collection_type=''*, *collection_table=''*, *output_mode='dataframe'*)

A method for fetching information from Collection, File, Collection_group tables

Parameters

- `collection_name` – A collection name to fetch the linked files
- `collection_type` – A collection type
- `collection_table` – A collection table
- `output_mode` – dataframe / object

`load_file_and_create_collection`(*data*, *autosave=True*, *hasher='md5'*, *calculate_file_size_and_md5=True*, *required_coumns=('name', 'type', 'table', 'file_path', 'size', 'md5', 'location')*)

A function for loading files to db and creating collections

Parameters

- `data` – A list of dictionary or a Pandas dataframe
- `autosave` – Save data to db, default True
- `required_coumns` – List of required columns
- `hasher` – Method for file checksum, default md5
- `calculate_file_size_and_md5` – Enable file size and md5 check, default True

Returns
None

`static prepare_data_for_collection_attribute`(*collection_name*, *collection_type*, *data_list*)

A static method for building data structure for collection attribute table update

Parameters

- `collection_name` – A collection name
- `collection_type` – A collection type
- `data` – A list of dictionaries containing the data for attribute table

Returns
A new list of dictionary for the collection attribute table

`remove_collection_group_info`(*data*, *autosave=True*, *required_collection_column=('name', 'type')*, *required_file_column='file_path'*)

A method for removing collection group information from database

Parameters

- `data` – A list dictionary or a Pandas DataFrame with following columns
  - name
  - type
  - file_path

  File_path information is not mandatory
- `required_collection_column` – List of required column for fetching collection, default 'name','type'
- `required_file_column` – Required column for fetching file information, default file_path
- `autosave` – A toggle for saving changes to database, default True

`store_collection_and_attribute_data`(*data*, *autosave=True*)

A method for dividing and storing data to collection and attribute table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `autosave` – A toggle for saving changes to database, default True

Returns
None

`store_collection_attributes`(*data*, *collection_id=''*, *autosave=False*)

A method for storing data to Collectionm_attribute table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `collection_id` – A collection id, optional
- `autosave` – A toggle for saving changes to database, default False

Returns
None

`store_collection_data`(*data*, *autosave=False*)

A method for loading data to Collection table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `autosave` – A toggle for saving changes to database, default True

Returns
None

`update_collection_attribute`(*collection_name*, *collection_type*, *attribute_name*, *attribute_value*, *autosave=True*)

A method for updating collection attribute

Parameters

- `collection_name` – A collection name
- `collection_type` – A collection type
- `attribute_name` – A collection attribute name
- `attribute_value` – A collection attribute value
- `autosave` – A toggle for committing changes to db, default True

File adaptor

`class igf_data.igfdb.fileadaptor.FileAdaptor`(***data*)

An adaptor class for File tables

`check_file_records_file_path`(*file_path*)

A method for checking file information in database

Parameters
`file_path` – A absolute filepath

Returns
True if the file is present in db or False if its not

`divide_data_to_table_and_attribute`(*data*, *required_column='file_path'*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for File and File_attribute tables

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `table_columns` – List of table column names, default None
- `required_column` – A column name to add to the attribute data
- `attribute_name_column` – A label for attribute name column
- `attribute_value_column` – A label for attribute value column

Returns
Two pandas dataframes, one for File and another for File_attribute table

`fetch_file_records_file_path`(*file_path*)

A method for fetching data for file table

Parameters
`file_path` – an absolute file path

Returns
A file object

`remove_file_data_for_file_path`(*file_path*, *remove_file=False*, *autosave=True*)

A method for removing entry for a specific file.

Parameters

- `file_path` – A complete file_path for checking database
- `remove_file` – A toggle for removing filepath, default False
- `autosave` – A toggle for automatically saving changes to database, default True

Returns
None

`store_file_and_attribute_data`(*data*, *autosave=True*)

A method for dividing and storing data to file and attribute table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `autosave` – A Toggle for automatically saving changes to db, default True

Returns
None

`store_file_attributes`(*data*, *file_id=''*, *autosave=False*)

A method for storing data to File_attribute table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `file_id` – A file_id for updating the attribute table, default empty string
- `autosave` – A Toggle for automatically saving changes to db, default True

Returns
None

`store_file_data`(*data*, *autosave=False*)

Load data to file table

Parameters

- `data` – A list of dictionary or a Pandas DataFrame
- `autosave` – A Toggle for automatically saving changes to db, default True

Returns
None

`update_file_table_for_file_path`(*file_path*, *tag*, *value*, *autosave=False*)

A method for updating file table

Parameters

- `file_path` – A file_path for database look up
- `tag` – A keyword for file column name
- `value` – A new value for the file column
- `autosave` – Toggle autosave, default off

Returns
None

Sequencing run adaptor

class igf_data.igfdb.seqrunadaptor.`SeqrunAdaptor`(***data*)

An adaptor class for table Seqrun

`check_seqrun_exists`(*seqrun_id*)

A method for checking if seqrun exists

Parameters
`seqrun_id` – Seqrun id

Returns
True if seqrun exists, False otherwise

`divide_data_to_table_and_attribute`(*data*, *required_column='seqrun_igf_id'*, *table_columns=None*, *attribute_name_column='attribute_name'*, *attribute_value_column='attribute_value'*)

A method for separating data for Seqrun and Seqrun_attribute tables

Parameters

- `data` – A list of dictionaries or a pandas dataframe
- `table_columns` – List of table column names, default None
- `required_column` – column name to add to the attribute data
- `attribute_name_column` – label for attribute name column
- `attribute_value_column` – label for attribute value column

Returns
two pandas dataframes, one for Seqrun and another for Run_attribute table

`fetch_flowcell_barcode_rules_for_seqrun`(*seqrun_igf_id*, *flowcell_label='flowcell'*, *output_mode='dataframe'*)

A method for fetching flowcell barcode rule for Seqrun

Parameters

- **seqrun_igf_id** – A seqrun igf id
- **flowcell_label** – Flowcell label, default 'flowcell'
- **output_mode** – Query output mode, default 'dataframe'

Returns
Flowcell rules records

`fetch_platform_info_for_seqrun`(*seqrun_igf_id*)
A method for fetching platform info for seqrun

Parameters
seqrun_igf_id – Seqrun igf id for platform name lookup

Returns
A string containing platform name or None

`fetch_seqrun_records_igf_id`(*seqrun_igf_id*, *target_column_name='seqrun_igf_id'*)
A method for fetching data for Seqrun table

Parameters

- **seqrun_igf_id** – an igf id
- **target_column_name** – a column name in the Seqrun table, default seqrun_igf_id

Returns
Seqrun record as oblect

`store_seqrun_and_attribute_data`(*data*, *autosave=True*)
A method for dividing and storing data to seqrun and attribute table

Parameters

- **data** – A list of dictionary or a Pandas dataframe containing Seqrun data
- **autosave** – A toggle for auto commit, default True

Returns
None

`store_seqrun_attributes`(*data*, *seqrun_id=''*, *autosave=False*)
A method for storing data to Seqrun_attribute table

Parameters

- **data** – A list of dictionary or a Pandas dataframe containing Seqrun attribute data
- **autosave** – A toggle for auto commit, default True

Returns
None

`store_seqrun_data`(*data*, *autosave=False*)
Load data to Seqrun table

Parameters

- **data** – A list of dictionary or a Pandas dataframe containing Seqrun data
- **autosave** – A toggle for auto commit, default True

Returns
None

`store_seqrun_stats_data`(*data*, *seqrun_id=''*, *autosave=True*)
A method for storing data to seqrun_stats table

Parameters

- **data** – A list of dictionary or a Pandas dataframe containing Seqrun stats data
- **seqrun_id** – Seqrun id info, default ‘’
- **autosave** – A toggle for auto commit, default True

Returns
None

Platform adaptor

class igf_data.igfdb.platformadaptor.**PlatformAdaptor**(***data*)

An adaptor class for Platform tables

fetch_platform_records_igf_id(*platform_igf_id*, *target_column_name='platform_igf_id'*, *output_mode='one'*)

A method for fetching data for Platform table

Parameters

- **platform_igf_id** – an igf id
- **target_column_name** – column name in the Platform table, default is platform_igf_id

Returns
Platform record as object

store_flowcell_barcode_rule(*data*, *autosave=True*)

Load data to flowcell_barcode_rule table

Parameters

- **data** – A dictionary or dataframe containing following columns
  - platform_igf_id / platform_id
  - flowcell_type
  - index_1 (NO_CHANGE/REVCOMP/UNKNOWN)
  - index_2 (NO_CHANGE/REVCOMP/UNKNOWN)
- **autosave** – A toggle for autocommit, default True

Returns
None

store_platform_data(*data*, *autosave=True*)

Load data to Platform table

Parameters

- **data** – A list of dictionaries or a pandas dataframe
- **autosave** – A toggle for autocommit, default False

Returns
None

Pipeline adaptor

class `igf_data.igfdb.pipelineadaptor.`**`PipelineAdaptor`**(***data*)

An adaptor class for Pipeline and Pipeline_seed tables

`check_seed_id_status`(*seed_id: str*, *seed_table: str*) → list

A method for checking the status of a seed_id in pipeline_seed table

Parameters

- **`seed_id`** – A string, seed_id
- **`seed_table`** – A string, seed_table

Returns

A list of strings, status of the seed_id

`create_pipeline_seed`(*data*, *autosave=True*, *status_column='status'*, *seeded_label='SEEDED'*, *required_columns=('pipeline_id', 'seed_id', 'seed_table')*)

A method for creating new entry in th pipeline_seed table

Parameters

- **`data`** – Dataframe or hash, it sould contain following fields
  - pipeline_name / pipeline_id
  - seed_id
  - seed_table
- **`autosave`** – A toggle for autocommit, default True
- **`status_column`** – Status column name, default status
- **`seeded_label`** – Seeded state label, default SEEDED
- **`required_columns`** – List of required columns, deafult [pipeline_id,seed_id,seed_table]

Returns

None

`fetch_pipeline_records_pipeline_name`(*pipeline_name*, *target_column_name='pipeline_name'*)

A method for fetching data for Pipeline table

Parameters

- **`pipeline_name`** – a name
- **`target_column_name`** – default pipeline_name

Returns

Pipeline record

`fetch_pipeline_seed`(*pipeline_id*, *seed_id*, *seed_table*, *target_column_name=('pipeline_id', 'seed_id', 'seed_table')*)

A method for fetching unique pipeline seed using pipeline_id, seed_id and seed_table

Parameters

- **`pipeline_id`** – A pipeline db id
- **`seed_id`** – A seed entry db id
- **`seed_table`** – A seed table name
- **`target_column_name`** – Target set of columns

Returns

Pipeline seed records

`fetch_pipeline_seed_with_table_data`(*pipeline_name*, *table_name='seqrun'*, *status='SEEDED'*)

A method for fetching linked table records for the seeded entries in pipeseed table

Parameters

- `pipeline_name` – A pipeline name
- `table_name` – A table name for pipeline_seed lookup, default seqrun
- `status` – A text label for seeded status, default is SEEDED

Returns

Two pandas dataframe for pipeline_seed entries and data from other tables

`seed_new_experiments`(*pipeline_name*, *species_name_list*, *fastq_type*, *project_list=None*, *library_source_list=None*, *active_status='ACTIVE'*, *autosave=True*, *seed_table='experiment'*)

A method for seeding new experiments for primary analysis

Parameters

- `pipeline_name` – Name of the analysis pipeline
- `project_list` – List of projects to consider for seeding analysis pipeline, default None
- `library_source_list` – List of library source to consider for analysis, default None
- `species_name_list` – List of sample species to consider for seeding analysis pipeline
- `active_status` – Label for active status, default ACTIVE
- `autosave` – A toggle for autosaving records in database, default True
- `seed_tabel` – Seed table for pipeseed table, default experiment

Returns

A list of available projects for seeding analysis table (if project_list is None) or None and a list of seeded experiments or None

`seed_new_seqruns`(*pipeline_name*, *autosave=True*, *seed_table='seqrun'*)

A method for creating seed for new seqruns

Parameters

- `pipeline_name` – A pipeline name
- `autosave` – A toggle for autocommit, default True
- `seed_table` – Seed table name, default seqrun

Returns

None

`store_pipeline_data`(*data*, *autosave=True*)

Load data to Pipeline table

Parameters

- `data` – A list of dictionaries or a Pandas dataframe
- `autosave` – A toggle for autocommit, default True

Returns

None

`update_pipeline_seed`(*data*, *autosave=True*, *required_columns=('pipeline_id', 'seed_id', 'seed_table', 'status')*)

A method for updating the seed status in pipeline_seed table

Parameters

- **data** – dataframe or a dictionary, should contain following fields
  - pipeline_name / pipeline_id
  - seed_id
  - seed_table
  - status
- **autosave** – A toggle for autocommit, default True
- **required_columns** – A list of required columns, default [pipeline_id,seed_id,seed_table,status]

Returns
None

2.1.3 Utility functions for database access

Database utility functions

igf_data.utils.dbutils.**clean_and_rebuild_database**(*dbconfig*)

A method for deleting data in database and create empty tables

Parameters
dbconfig – A json file containing the database connection info

igf_data.utils.dbutils.**read_dbconf_json**(*dbconfig*)

A method for reading dbconfig json file

Parameters
dbconfig – A json file containing the database connection info e.g. {"dbhost":"DBHOST","dbport": PORT,"dbuser":"USER","dbpass":"DBPASS","dbname":"DBNAME","driver":"myso

Returns
a dictionary containing dbparms

igf_data.utils.dbutils.**read_json_data**(*data_file*)

A method for reading data from json file

Parameters
data_file – A Json format file

Returns
A list of dictionaries

Project adaptor utility functions

igf_data.utils.projectutils.**draft_email_for_project_cleanup**(*template_file*, *data*, *draft_output*)

A method for drafting email for cleanup

Parameters

- **template_file** – A template file
- **data** – A list of dictionary or a dictionary containing the following columns
  - name
  - email_id
  - projects
  - cleanup_date

- **draft_output** – A output filename

`igf_data.utils.projectutils.`**`find_projects_for_cleanup`**(*dbconfig_file*, *warning_note_weeks=24*, *all_warning_note=False*)

A function for finding old projects for cleanup

Parameters

- **dbconfig_file** – A dbconfig file path
- **warning_note_weeks** – Number of weeks from last sequencing run to wait before sending warnings, default 24
- **all_warning_note** – A toggle for sending warning notes to all, default False

Returns

A list containing warning lists, a list containing final note list and another list with clean up list

`igf_data.utils.projectutils.`**`get_files_and_irods_path_for_project`**(*project_igf_id*, *db_session_class*, *irods_path_prefix='/igfZone/home/'*)

A function for listing all the files and irods dir path for a given project

Parameters

- **project_igf_id** – A string containing the project igf id
- **db_session_class** – A database session object
- **irods_path_prefix** – A string containing irods path prefix, default '/igfZone/home/'

Returns

A list containing all the files for a project and a string containing the irods path for the project

`igf_data.utils.projectutils.`**`get_project_read_count`**(*project_igf_id: str*, *session_class: Optional[Any] = None*, *dbconfig_file: Optional[Any] = None*, *run_attribute_name: str = 'R1_READ_COUNT'*, *active_status: str = 'ACTIVE'*) → DataFrame

A utility method for fetching sample read counts for an input project_igf_id

Parameters

- **project_igf_id** – A project_igf_id string
- **session_class** – A db session class object, default None
- **dbconfig_file** – A db config file, default None
- **run_attribute_name** – Attribute name from Run_attribute table for read count lookup
- **active_status** – text label for active runs, default ACTIVE

Returns

A pandas dataframe containing following columns

- project_igf_id
- sample_igf_id
- flowcell_id
- attribute_value

`igf_data.utils.projectutils.`**`get_seqrun_info_for_project`**(*project_igf_id: str*, *session_class: Optional[Any] = None*, *dbconfig_file: Optional[Any] = None*) → DataFrame

A utility method for fetching seqrun_igf_id and flowcell_id which are linked to a specific project_igf_id

Parameters

- **project_igf_id** – A project_igf_id string
- **session_class** – A db session class object, default None
- **dbconfig_file** – A db config file, default None

Returns

A pandas dataframe containing following columns

- seqrun_igf_id
- flowcell_id

igf_data.utils.projectutils.**mark_project_and_list_files_for_cleanup**(*project_igf_id*, *dbconfig_file*, *outout_dir*, *force_overwrite=True*, *use_ephemeral_space=False*, *irods_path_prefix='/igfZone/home/'*, *withdrawn_tag='WITHDRAWN'*)

A wrapper function for project cleanup operation

Parameters

- **project_igf_id** – A string of project igf -id
- **dbconfig_file** – A dbconf json file path
- **outout_dir** – Output dir path for dumping file lists for project
- **force_overwrite** – Overwrite existing output file, default True
- **use_ephemeral_space** – A toggle for temp dir, default False
- **irods_path_prefix** – Prefix for irods path, default /igfZone/home/
- **withdrawn_tag** – A string tag for marking files in db, default WITHDRAWN

Returns
None

igf_data.utils.projectutils.**mark_project_as_withdrawn**(*project_igf_id*, *db_session_class*, *withdrawn_tag='WITHDRAWN'*)

A function for marking all the entries for a specific project as withdrawn

Parameters

- **project_igf_id** – A string containing the project igf id
- **db_session_class** – A dbsession object
- **withdrawn_tag** – A string for withdrawn field in db, default WITHDRAWN

Returns
None

igf_data.utils.projectutils.**mark_project_barcode_check_off**(*project_igf_id*, *session_class*, *barcode_check_attribute='barcode_check'*, *barcode_check_val='OFF'*)

A utility method for marking project barcode check as off using the project_igf_id

Parameters

- **project_igf_id** – A project_igf_id string
- **session_class** – A db session class object

- **barcode_check_attribute** – A text keyword for barcode check attribute, default barcode_check
- **barcode_check_val** – A text for barcode check attribute value, default is 'OFF'

Returns
None

igf_data.utils.projectutils.**notify_project_for_cleanup**(*warning_template*, *final_notice_template*, *cleanup_template*, *warning_note_list*, *final_note_list*, *cleanup_list*, *use_ephemeral_space=False*)

A function for sending emails to users for project cleanup

Parameters

- **warning_template** – A email template file for warning
- **final_notice_template** – A email template for final notice
- **cleanup_template** – A email template for sending cleanup list to igf
- **warning_note_list** – A list of dictionary containing following fields to warn user about cleanup
  - name
  - email_id
  - projects
  - cleanup_date
- **final_note_list** – A list of dictionary containing above mentioned fields to noftify user about final cleanup
- **cleanup_list** – A list of dictionary containing above mentioned fields to list projects for cleanup
- **use_ephemeral_space** – A toggle for using the ephemeral space, default False

igf_data.utils.projectutils.**send_email_to_user_via_sendmail**(*draft_email_file*, *waiting_time=20*, *sendmail_exe='sendmail'*, *dry_run=False*)

A function for sending email to users via sendmail

Parameters

- **draft_email_file** – A draft email to be sent to user
- **waiting_time** – Wait after sending the email, default 20sec
- **sendmail_exe** – Sendmail exe path, default sendmail
- **dry_run** – A toggle for dry run, default False

Sequencing adaptor utility functions

`igf_data.utils.seqrunutils.`**`get_seqrun_date_from_igf_id`**`(`*seqrun_igf_id: str*`)` → str

A utility method for fetching sequence run date from the igf id

Params seqrun_igf_id
A seqrun igf id string

Returns
a string value of the date

`igf_data.utils.seqrunutils.`**`load_new_seqrun_data`**`(`*data_file: str*, *dbconfig: str*`)` → None

A method for loading new data for seqrun table

Pipeline adaptor utility functions

`igf_data.utils.pipelineutils.`**`find_new_analysis_seeds`**`(`*dbconfig_path*, *pipeline_name*, *project_name_file*, *species_name_list*, *fastq_type*, *library_source_list*`)`

A utils method for finding and seeding new experiments for analysis

Parameters

- **`dbconfig_path`** – A database configuration file
- **`slack_config`** – A slack configuration file

:param pipeline_name:Pipeline name :param fastq_type: Fastq collection type :param project_name_file: A file containing the list of projects for seeding pipeline :param species_name_list: A list of species to consider for seeding analysis :param library_source_list: A list of library source info to consider for seeding analysis :returns: List of available experiments or None and a list of seeded experiments or None

`igf_data.utils.pipelineutils.`**`load_new_pipeline_data`**`(`*data_file*, *dbconfig*`)`

A method for loading new data for pipeline table

Platform adaptor utility functions

`igf_data.utils.platformutils.`**`load_new_flowcell_data`**`(`*data_file*, *dbconfig*`)`

A method for loading new data to flowcell table

`igf_data.utils.platformutils.`**`load_new_platform_data`**`(`*data_file*, *dbconfig*`)`

A method for loading new data for platform table

Pipeline seed adaptor utility functions

`igf_data.utils.ehive_utils.pipeseedfactory_utils.`**`get_pipeline_seeds`**`(`*pipeseed_mode*, *pipeline_name*, *igf_session_class*, *seed_id_label='seed_id'*, *seqrun_date_label='seqrun_date'*, *seqrun_id_label='seqrun_id'*, *experiment_id_label='experiment_id'*, *seqrun_igf_id_label='seqrun_igf_id'*`)`

A utils function for fetching pipeline seed information

Parameters

- **pipeseed_mode** – A string info about pipeseed mode, allowed values are demultiplexing alignment
- **pipeline_name** – A string infor about pipeline name
- **igf_session_class** – A database session class for pipeline seed lookup

Returns
Two Pandas dataframes, first with pipeseed entries and second with seed info

2.2 IGF pipeline api

2.2.1 Pipeline api

Fetch fastq files for analysis

igf_data.utils.analysis_fastq_fetch_utils.**get_fastq_and_run_for_samples**(*dbconfig_file, sample_igf_id_list, active_status='ACTIVE', combine_fastq_dir=False, fastq_collection_type='demultiplexed*

A function for fetching fastq and run_igf_id for a list od samples

Parameters

- **dbconfig_file** – DB config file path
- **sample_igf_id_list** – A list of sample_igf_ids for DB lookup
- **active_status** – Filter tag for active experiment, run and file status, default: active
- **combine_fastq_dir** – Combine fastq file directories for output line, default False
- **fastq_collection_type** – Fastq collection type, default 'demultiplexed_fastq'

Returns
A list of dictionary containing the sample_igf_id, run_igf_id and file_paths

igf_data.utils.analysis_fastq_fetch_utils.**get_fastq_input_list**(*db_session_class, experiment_igf_id, combine_fastq_dir=False, fastq_collection_type='demultiplexed_fastq', active_status='ACTIVE'*)

A function for fetching all the fastq files linked to a specific experiment id

Parameters

- **db_session_class** – A database session class
- **experiment_igf_id** – An experiment igf id
- **fastq_collection_type** – Fastq collection type name, default demultiplexed_fastq
- **active_status** – text label for active runs, default ACTIVE
- **combine_fastq_dir** – Combine fastq file directories for output line, default False

Returns
A list of fastq file or fastq dir paths for the analysis run

Raises

`ValueError` – It raises ValueError if no fastq directory found

Load analysis result to database and file system

`class igf_data.utils.analysis_collection_utils.`**`Analysis_collection_utils`**(*dbsession_class, base_path=None, collection_name=None, collection_type=None, collection_table=None, rename_file=True, add_datestamp=True, tag_name=None, analysis_name=None, allowed_collection=('sample', 'experiment', 'run', 'project')*)

A class for dealing with analysis file collection. It has specific method for moving analysis files to a specific directory structure and rename the file using a uniform rule, if required. Example '<collection_name>_<analysis_name>_<tag>_<datestamp>.<original_suffix>'

Parameters

- **dbsession_class** – A database session class
- **collection_name** – Collection name information for file, default None
- **collection_type** – Collection type information for file, default None
- **collection_table** – Collection table information for file, default None
- **base_path** – A base filepath to move file while loading, default 'None' for no file move
- **rename_file** – Rename file based on collection_table type while loading, default True
- **add_datestamp** – Add datestamp while loading the file
- **analysis_name** – Analysis name for the file, required for renaming while loading, default None
- **tag_name** – Additional tag for filename,default None
- **allowed_collection** – List of allowed collection tables
  - sample
  - experiment
  - run
  - project

`create_or_update_analysis_collection`(*file_path, dbsession, withdraw_exisitng_collection=True, autosave_db=True, force=True, remove_file=False, size=None, md5=None, calculate_file_size_and_md5=True*)

A method for create or update analysis file collection in db. Required elements will be collected from database if base_path element is given.

Parameters

- `file_path` – file path to load as db collection
- `dbsession` – An active database session
- `withdraw_exisitng_collection` – Remove existing collection group
- `autosave_db` – Save changes to database, default True
- `remove_file` – A toggle for removing existing file from disk, default False
- `force` – Toggle for removing existing file collection, default True
- `size` – File size, default None
- `md5` – File md5, default None
- `calculate_file_size_and_md5` – Calculate file size and md5, default True

`get_new_file_name`(*input_file*, *file_suffix=None*)

A method for fetching new file name

Parameters

- `input_file` – An input filepath
- `file_suffix` – A file suffix

`load_file_to_disk_and_db`(*input_file_list*, *withdraw_exisitng_collection=True*, *path_label='file_path'*, *size_label='size'*, *md5_label='md5'*, *autosave_db=True*, *file_suffix=None*, *force=True*, *remove_file=False*)

A method for loading analysis results to disk and database. File will be moved to a new path if base_path is present. Directory structure of the final path is based on the collection_table information.

Following will be the final directory structure if base_path is present

- project - base_path/project_igf_id/analysis_name
- sample - base_path/project_igf_id/sample_igf_id/analysis_name
- experiment - base_path/project_igf_id/sample_igf_id/experiment_igf_id/analysis_name
- run - base_path/project_igf_id/sample_igf_id/experiment_igf_id/run_igf_id/analysis_name

Parameters

- `input_file_list` – A list of input file to load, all using the same collection info
- `withdraw_exisitng_collection` – Remove existing collection group, DO NOT use this while loading a list of files
- `autosave_db` – Save changes to database, default True
- `file_suffix` – Use a specific file suffix, use None if it should be same as original file e.g. input.vcf.gz to output.vcf.gz
- `path_label` – File path label: default file_path
- `size_label` – File size label, default size
- `md5_label` – File md5 label, default md5
- `force` – Toggle for removing existing file, default True
- `remove_file` – A toggle for removing existing file from disk, default False

Returns

A list of final filepath

Run metadata validation checks

`class igf_data.utils.validation_check.metadata_validation.`**`Validate_project_and_samplesheet_metadata`**`(`

A package for running validation checks for project and samplesheet metadata file

Parameters

- **`samplesheet_file`** – A samplesheet input file
- **`metadata_files`** – A list of metadata input file
- **`samplesheet_schema`** – A json schema for samplesheet file validation
- **`metadata_schema`** – A json schema for metadata file validation

`static` **`check_metadata_library_by_row`**`(`*data*`)`

A static method for checking library type metadata per row

Parameters
`data` – A pandas data series containing sample metadata

Returns
An error message or None

`compare_metadata``()`

A function for comparing samplesheet and metadata files

Returns
A list of error or an empty list

`convert_errors_to_gviz``(`*output_json=None*`)`

A method for converting the list of errors to gviz format json

Parameters
`output_json` – A output json file for saving data, default None

Returns
A gviz json data block for the html output if output_json is None, or else None

`dump_error_to_csv``(`*output_csv*`)`

A method for dumping list or errors to a csv file :returns: output csv file path if any errors found, or else None

`get_merged_errors``()`

A method for running the validation checks on input samplesheet metadata and samplesheet files :returns: A list of errors or an empty list

`get_metadata_validation_report``()`

A method for running validation check on input metdata files :returns: A list of errors or an empty list

`get_samplesheet_validation_report``()`

A method for running validation checks on input samplesheet file :returns: A list of errors or an empty list

`static` **`validate_metadata_library_type`**`(`*sample_id*, *library_source*, *library_strategy*, *experiment_type*`)`

A staticmethod for validating library metadata information for sample

Parameters

- **sample_id** – Sample name
- **library_source** – Library source information
- **library_strategy** – Library strategy information
- **experiment_type** – Experiment type information

Returns
A error message string or None

2.2.2 Generic utility functions

Basic fasta sequence processing

igf_data.utils.sequtils.**rev_comp**(*input_seq*)

A function for converting nucleotide sequence to its reverse complement

Parameters
input_seq – A string of nucleotide sequence

Returns
Reverse complement version of the input sequence

Advanced fastq file processing

igf_data.utils.fastq_utils.**compare_fastq_files_read_counts**(*r1_file*, *r2_file*)

A method for comparing read counts for fastq pairs

Parameters

- **r1_file** – Fastq pair R1 file path
- **r2_file** – Fastq pair R2 file path

Raises
ValueError if counts are not same

igf_data.utils.fastq_utils.**count_fastq_lines**(*fastq_file*)

A method for counting fastq lines

Parameters
fastq_file – A gzipped or unzipped fastq file

Returns
Fastq line count

igf_data.utils.fastq_utils.**detect_non_fastq_in_file_list**(*input_list*)

A method for detecting non fastq file within a list of input fastq

Parameters
input_list – A list of filepath to check

Returns
True in non fastq files are present or else False

igf_data.utils.fastq_utils.**identify_fastq_pair**(*input_list*, *sort_output=True*, *check_count=False*)

A method for fastq read pair identification

Parameters

- **input_list** – A list of input fastq files
- **sort_output** – Sort output list, default true

- **check_count** – Check read count for fastq pair, only available if sort_output is True, default False

Returns
A list for read1 files and another list of read2 files

Process local and remote files

igf_data.utils.fileutils.**calculate_file_checksum**(*filepath*, *hasher='md5'*)

A method for file checksum calculation

Parameters

- **filepath** – a file path
- **hasher** – default is md5, allowed: md5 or sha256

Returns
file checksum value

igf_data.utils.fileutils.**check_file_path**(*file_path*)

A function for checking existing filepath

Parameters
file_path – An input filepath for check

Raises
IOError – It raises IOError if file not found

igf_data.utils.fileutils.**copy_local_file**(*source_path: str*, *destination_path: str*, *cd_to_dest: bool = True*, *force: bool = False*, *dirs_exist_ok: bool = True*, *new_dir_mode: int = 504*) → None

A method for copy files to local disk

Parameters

- **source_path** – A source file path
- **destination_path** – A destination file path, including the file name ##FIX TYPO
- **cd_to_dest** – Change to destination dir before copy, default True
- **force** – Optional, set True to overwrite existing destination file, default is False
- **dirs_exist_ok** – Optional, set True to allow existing directories, default is False

Returns
None

igf_data.utils.fileutils.**copy_remote_file**(*source_path*, *destination_path*, *source_address=None*, *destination_address=None*, *ssh_key_file=None*, *copy_method='rsync'*, *check_file=True*, *force_update=False*, *exclude_pattern_list=None*)

A method for copy files from or to remote location

Parameters

- **source_path** – A source file path
- **destination_path** – A destination file path
- **source_address** – Address of the source server
- **destination_address** – Address of the destination server
- **ssh_key_file** – A path to the ssh key file, default None
- **copy_method** – A nethod for copy files, default is 'rsync'

- **check_file** – Check file after transfer using checksum, default True
- **force_update** – Overwrite existing file or dir, default is False
- **exclude_pattern_list** – List of file pattern to exclude, Deefault None

igf_data.utils.fileutils.**create_file_manifest_for_dir**(*results_dirpath*, *output_file*, *md5_label='md5'*, *size_lavel='size'*, *path_label='file_path'*, *exclude_list=None*, *force=True*)

A method for creating md5 and size list for all the files in a directory path

Parameters

- **results_dirpath** – A file path for input file directory
- **output_file** – Name of the output csv filepath
- **exclude_list** – A list of file pattern to exclude from the archive, default None
- **force** – A toggle for replacing output file, if its already present, default True
- **md5_label** – A string for checksum column, default md5
- **size_lavel** – A string for file size column, default size
- **path_label** – A string for file path column, default file_path

Returns
Nill

igf_data.utils.fileutils.**get_date_stamp**()

A method for generating datestamp for files

Returns
A string of datestampe in 'YYYY-MM-DD HH:MM' format

igf_data.utils.fileutils.**get_date_stamp_for_file_name**() → str

A method for generating datestamp for file name

Returns
A string of datestampe in 'YYYY_MM_DD_HH_MM' format

igf_data.utils.fileutils.**get_datestamp_label**(*datetime_str=None*)

A method for fetching datestamp

Parameters
datetime_str – A datetime string to parse, default None

Returns
A padded string of format YYYYMMDD

igf_data.utils.fileutils.**get_file_extension**(*input_file*)

A method for extracting file suffix information

Parameters
input_file – A filepath for getting suffix

Returns
A suffix string or an empty string if no suffix found

igf_data.utils.fileutils.**get_temp_dir**(*work_dir=None*, *prefix='temp'*, *use_ephemeral_space=False*)

A function for creating temp directory

Parameters

- **work_dir** – A path for work directory, default None
- **prefix** – A prefix for directory path, default 'temp'

- `use_ephemeral_space` – Use env variable $EPHEMERAL to get work directory, default False

Returns
A temp_dir

`igf_data.utils.fileutils.`**`list_remote_file_or_dirs`**(*remote_server*, *remote_path*, *only_dirs=True*, *only_files=False*, *user_name=None*, *user_pass=None*)

A method for listing dirs or files on the remote dir paths

Parameters

- `remote_server` – Semote servet address
- `remote_path` – Path on remote server
- `only_dirs` – Toggle for listing only dirs, default True
- `only_files` – Toggle for listing only files, default False
- `user_name` – User name, default None
- `user_pass` – User pass, default None

Returns
A list of dir or file paths

`igf_data.utils.fileutils.`**`move_file`**(*source_path*, *destination_path*, *cd_to_dest=False*, *force=False*)

A method for moving files to local disk

Parameters

- `source_path` – A source file path
- `destination_path` – A destination file path, including the file name
- `force` – Optional, set True to overwrite existing destination file, default is False

`igf_data.utils.fileutils.`**`prepare_file_archive`**(*results_dirpath*, *output_file*, *gzip_output=True*, *exclude_list=None*, *force=True*, *output_mode='w'*)

A method for creating tar.gz archive with the files present in filepath

Parameters

- `results_dirpath` – A file path for input file directory
- `output_file` – Name of the output archive filepath
- `gzip_output` – A toggle for creating gzip output tarfile, default True
- `exclude_list` – A list of file pattern to exclude from the archive, default None
- `force` – A toggle for replacing output file, if its already present, default True
- `output_mode` – File output mode, default 'w' (or 'w:gz: for gzip files)

Returns
None

`igf_data.utils.fileutils.`**`preprocess_path_name`**(*input_path*)

A method for processing a filepath. It takes a file path or dirpath and returns the same path after removing any whitespace or ascii symbols from the input.

Parameters
`path` – An input file path or directory path

Returns
A reformatted filepath or dirpath

`igf_data.utils.fileutils.`**`read_json_data`**`(`*data_file*`)`

A method for reading data from json file

Parameters
`data_file` – A Json format file

Returns
A list of dictionaries

`igf_data.utils.fileutils.`**`remove_dir`**`(`*dir_path*, *ignore_errors=True*`)`

A function for removing directory containing files

Parameters

- **`dir_path`** – A directory path
- **`ignore_errors`** – Ignore errors while removing dir, default True

Load files to irods server

`class igf_data.utils.igf_irods_client.`**`IGF_irods_uploader`**`(`*irods_exe_dir*, *host='eliot.med.ic.ac.uk'*, *zone='/igfZone'*, *port=1247*, *igf_user='igf'*, *irods_resource='woolfResc'*`)`

A simple wrapper for uploading files to irods server from HPC cluster CX1 Please run the following commands in the HPC cluster before running this module Add irods settings to ~/.irods/irods_environment.json > module load irods/4.2.0 > iinit (optional username) Authenticate irods settings using your password The above command will generate a file containing your iRODS password in a 'scrambled form'

Parameters
`irods_exe_dir` – A path to the bin directory where icommands are installed

`upload_analysis_results_and_create_collection``(`*file_list*, *irods_user*, *project_name*, *analysis_name='default'*, *dir_path_list=None*, *file_tag=None*`)`

A method for uploading analysis files to irods server

Parameters

- **`file_list`** – A list of file paths to upload to irods
- **`irods_user`** – Irods user name
- **`project_name`** – Name of the project_name
- **`analysis_name`** – A string for analysis name, default is 'default'
- **`dir_path_list`** – A list of directory structure for irod server, default None for using datestamp
- **`file_tag`** – A text string for adding tag to collection, default None for only project_name

`upload_fastqfile_and_create_collection``(`*filepath*, *irods_user*, *project_name*, *run_igf_id*, *run_date*, *flowcell_id=None*, *data_type='fastq'*`)`

A method for uploading files to irods server and creating collections with metadata

Parameters

- **`filepath`** – A file for upload to iRODS server
- **`irods_user`** – Recipient user's irods username
- **`project_name`** – Name of the project. This will be user for collection tag

- **run_igf_id** – A unique igf id, either seqrun or run or experiment
- **run_date** – A unique run date
- **data_type** – A directory label, e.g, fastq, bam or cram

Calculate storage statistics

igf_data.utils.disk_usage_utils.**get_storage_stats_in_gb**(*storage_list*)

A utility function for fetching disk usage stats (df -h) and return disk usge in Gb

Parameters
storage_list – a input list of storage path

Returns

A list of dictionary containing following keys

storage used available

igf_data.utils.disk_usage_utils.**get_sub_directory_size_in_gb**(*input_path*, *dir_name_col='directory_name'*, *dir_size_col='directory_size'*)

A utility function for listing disk size of all sub-directories for a given path (similar to linux command du -sh /path/*)

Parameters

- **input_path** – a input file path
- **dir_name_col** – column name for directory name, default directory_name
- **dir_size_col** – column name for directory size, default directory size

Returns

- **a list of dictionaries containing following keys**
  directory_name directory_size
- a description dictionary for gviz_api
- a column order list for gviz _api

igf_data.utils.disk_usage_utils.**merge_storage_stats_json**(*config_file*, *label_file=None*, *server_name_col='server_name'*, *storage_col='storage'*, *used_col='used'*, *available_col='available'*, *disk_usage_col='disk_usage'*)

A utility function for merging multiple disk usage stats file generated by json dump of get_storage_stats_in_gb output

Parameters

- **config_file** – a disk usage status config json file with following keys

  server_name disk_usage

  Each of the disk usage json files should have following keys

  storage used available

- **label_file** – an optional json file for renaming the raw disk names format: <raw name> : <print name>

Returns

- merged data as a list of dictionaries

- a dictionary containing the description for the gviz_data
- a list of column order

2.2.3 Run analysis tools

Process fastqc output file

igf_data.utils.fastqc_utils.**get_fastq_info_from_fastq_zip**(*fastqc_zip*, *fastqc_datafile='*/fastqc_data.txt'*)

A function for retriving total reads and fastq file name from fastqc_zip file

Parameters

- **fastqc_zip** – A zip file containing fastqc results
- **fastqc_datafile** – A pattern f

Returns

return total read count and fastq filename

Cellranger count utils

igf_data.utils.tools.cellranger.cellranger_count_utils.**check_cellranger_count_output**(*output_path*, *file_list=('web_sum 'met- rics_summary.csv', 'pos- sorted_genome_ban 'pos- sorted_genome_ban 'fil- tered_feature_bc_m 'raw_feature_bc_ma 'molecule_info.h5', 'cloupe.cloupe', 'anal- y- sis/tsne/2_compone 'anal- y- sis/clustering/graph 'anal- y- sis/diffexp/kmeans_ 'anal- y- sis/pca/10_compone*

A function for checking cellranger count output

Parameters

- **output_path** – A filepath for cellranger count output directory
- **file_list** – List of files to check in the output directory

  default file list to check

  – web_summary.html
  – metrics_summary.csv

- possorted_genome_bam.bam
- possorted_genome_bam.bam.bai
- filtered_feature_bc_matrix.h5
- raw_feature_bc_matrix.h5
- molecule_info.h5
- cloupe.cloupe
- analysis/tsne/2_components/projection.csv
- analysis/clustering/graphclust/clusters.csv
- analysis/diffexp/kmeans_3_clusters/differential_expression.csv
- analysis/pca/10_components/variance.csv

Returns
None

Raises
`IOError` – when any file is missing from the output path

`igf_data.utils.tools.cellranger.cellranger_count_utils.`**`extract_cellranger_count_metrics_summary`**(*cellra col- lec- tion_ col- lec- tion_ at- tribut at- tribut at- tribut tar- get_fi*

A function for extracting metrics summary file for cellranger ourput tar and parse the file. Optionally it can add the collection name and type info to the output dictionary.

Parameters

- **`cellranger_tar`** – A cellranger output tar file
- **`target_filename`** – A filename for metrics summary file lookup, default metrics_summary.csv
- **`collection_name`** – Optional collection name, default None
- **`collection_type`** – Optional collection type, default None
- **`attribute_tag`** – An optional string to add as prefix of the attribute names, default None

Returns
A dictionary containing the metrics values

`igf_data.utils.tools.cellranger.cellranger_count_utils.`**`get_cellranger_count_input_list`**`(`*db_session_class ex- per- i- ment_igf_id*, *fastq_collection_ ac- tive_status='ACT*

A function for fetching input list for cellranger count run for a specific experiment

Parameters

- **`db_session_class`** – A database session class
- **`experiment_igf_id`** – An experiment igf id
- **`fastq_collection_type`** – Fastq collection type name, default demultiplexed_fastq
- **`active_status`** – text label for active runs, default ACTIVE

Returns
A list of fastq dir path for the cellranger count run

Raises
`ValueError` – It raises ValueError if no fastq directory found

`igf_data.utils.tools.cellranger.cellranger_count_utils.`**`run_cellranger_multi`**`(`*cellranger_exe*, *library_csv*, *sample_id*, *output_dir*, *use_ephemeral_space=False*, *job_timeout=43200*, *cellranger_options=('- -localcores 1', '--localmem 8')*`)`

A function for running Cellranger multi tool

Parameters

- **`cellranger_exe`** – Path to Cellranger exe
- **`library_csv`** – Path to library.csv file
- **`sample_id`** – Sample id
- **`output_dir`** – Output path
- **`use_ephemeral_space`** – A toggle for using EPHEMERAL temp space, default False
- **`job_timeout`** – Job timeout, default 43200
- **`cellranger_options`** – Cellranger multi options, default: ('–localcores 1','–localmem 8')

Returns
two strings, cmd and output_dir

BWA utils

class igf_data.utils.tools.bwa_utils.**BWA_util**(*bwa_exe*, *samtools_exe*, *ref_genome*, *input_fastq_list*, *output_dir*, *output_prefix*, *bam_output=True*, *thread=1*, *use_ephemeral_space=0*)

Pipeline utils class for running BWA

Parameters

- **bwa_exe** – BWA executable path
- **samtools_exe** – Samtools executable path
- **ref_genome** – Reference genome index for BWA run
- **input_fastq_list** – List of input fastq files for alignment
- **output_dir** – Output directory path
- **output_prefix** – Output prefix for alignment
- **bam_output** – A toggle for writing bam output, default True
- **thread** – No. of threads for BWA run, default 1
- **use_ephemeral_space** – A toggle for temp dir settings, default 0

run_mem(*mem_cmd='mem'*, *parameter_options=('-M', '')*, *samtools_cmd='view'*, *dry_run=False*)

A method for running Bwa mem and generate output alignment

Parameters

- **mem_cmd** – Bwa mem command, default mem
- **option_list** – List of bwa mem option, default -M
- **samtools_cmd** – Samtools view command, default view
- **dry_run** – A toggle for returning the bwa cmd without running it, default False

Returns

A alignment file path and bwa run cmd

Picard utils

class igf_data.utils.tools.picard_util.**Picard_tools**(*java_exe*, *picard_jar*, *input_files*, *output_dir*, *ref_fasta*, *picard_option=None*, *java_param='-Xmx4g'*, *strand_info='NONE'*, *threads=1*, *output_prefix=None*, *use_ephemeral_space=0*, *ref_flat_file=None*, *ribisomal_interval=None*, *patterned_flowcell=False*, *singularity_image=None*, *supported_commands=('CollectAlignmentSummaryMetrics', 'CollectGcBiasMetrics', 'QualityScoreDistribution', 'CollectRnaSeqMetrics', 'CollectBaseDistributionByCycle', 'MarkDuplicates', 'AddOrReplaceReadGroups')*)

A class for running picard tool

Parameters

- **java_exe** – Java executable path
- **picard_jar** – Picard path
- **input_files** – Input bam filepaths list
- **output_dir** – Output directory filepath
- **ref_fasta** – Input reference fasta filepath
- **picard_option** – Additional picard run parameters as dictionary, default None
- **java_param** – Java parameter, default '-Xmx4g'
- **strand_info** – RNA-Seq strand information, default NONE
- **ref_flat_file** – Input ref_flat file path, default None
- **output_prefix** – Output prefix name, default None
- **threads** – Number of threads to run for java, default 1
- **use_ephemeral_space** – A toggle for temp dir setting, default 0
- **patterned_flowcell** – Toggle for marking the patterned flowcell, default False
- **singularity_image** – Singularity image path, default None
- **suported_commands** – A list of supported picard commands
  - CollectAlignmentSummaryMetrics
  - CollectGcBiasMetrics
  - QualityScoreDistribution
  - CollectRnaSeqMetrics
  - CollectBaseDistributionByCycle
  - MarkDuplicates
  - AddOrReplaceReadGroups

run_picard_command(*command_name*, *dry_run=False*)

A method for running generic picard command

Parameters

- **command_name** – Picard command name
- **dry_run** – A toggle for returning picard command without the actual run, default False

Returns

A list of output files from picard run and picard run command and optional picard metrics

Fastp utils

class igf_data.utils.tools.fastp_utils.**Fastp_utils**(*fastp_exe*, *input_fastq_list*, *output_dir*, *run_thread=1*, *enable_polyg_trim=False*, *split_by_lines_count=5000000*, *log_output_prefix=None*, *use_ephemeral_space=0*, *fastp_options_list=('-a', 'auto', '--qualified_quality_phred=15', '--length_required=15')*)

A class for running fastp tool for a list of input fastq files

Parameters

- **fastp_exe** – A fastp executable path
- **input_fastq_list** – A list of input files
- **output_dir** – A output directory path
- **split_by_lines_count** – Number of entries for splitted fastq files, default 5000000
- **run_thread** – Number of threads to use, default 1
- **enable_polyg_trim** – Enable poly G trim for NextSeq and NovaSeq, default False
- **log_output_prefix** – Output prefix for log file, default None
- **use_ephemeral_space** – A toggle for temp dir, default 0
- **fastp_options_list** – A list of options for running fastp, default -a auto –qualified_quality_phred 15 –length_required=15

run_adapter_trimming(*split_fastq=False*, *force_overwrite=True*)

A method for running fastp adapter trimming

Parameters
split_fastq – Split fastq output files by line counts, default False

Pram force_overwrite
A toggle for overwriting existing file, default True

Returns
A list for read1 files, list of read2 files and a html report file and the fastp commandline

GATK utils

class igf_data.utils.tools.gatk_utils.**GATK_tools**(*gatk_exe*, *ref_fasta*, *use_ephemeral_space=False*, *java_param='-XX:ParallelGCThreads=1 -Xmx4g'*)

A python class for running gatk tools

Parameters

- **gatk_exe** – Gatk exe path
- **java_param** – Java parameter, default '-XX:ParallelGCThreads=1 -Xmx4g'
- **ref_fasta** – Input reference fasta filepath
- **use_ephemeral_space** – A toggle for temp dir settings, default False

run_AnalyzeCovariates(*before_report_file*, *after_report_file*, *output_pdf_path*, *force=False*, *dry_run=False*, *gatk_param_list=None*)

A method for running GATK AnalyzeCovariates tool

Parameters

- **before_report_file** – A file containing bqsr output before recalibration
- **after_report_file** – A file containing bqsr output after recalibration
- **output_pdf_path** – An output pdf filepath
- **force** – Overwrite output file, if force is True
- **dry_run** – Return GATK command, if its true, default False
- **gatk_param_list** – List of additional params for BQSR, default None

Returns
GATK commandline

run_ApplyBQSR(*bqsr_recal_file*, *input_bam*, *output_bam_path*, *force=False*, *dry_run=False*, *gatk_param_list=None*)

A method for running GATK ApplyBQSR

Parameters

- **input_bam** – An input bam file
- **bqsr_recal_file** – An bqsr table filepath
- **output_bam_path** – A bam output file
- **force** – Overwrite output file, if force is True
- **dry_run** – Return GATK command, if its true, default False
- **gatk_param_list** – List of additional params for BQSR, default None

Returns
GATK commandline

run_BaseRecalibrator(*input_bam*, *output_table*, *known_snp_sites=None*, *known_indel_sites=None*, *force=False*, *dry_run=False*, *gatk_param_list=None*)

A method for running GATK BaseRecalibrator

Parameters

- **input_bam** – An input bam file
- **output_table** – An output table filepath for recalibration results
- **known_snp_sites** – Known snp sites (e.g. dbSNP vcf file), default None
- **known_indel_sites** – Known indel sites (e.g.Mills_and_1000G_gold_standard indels vcf), default None
- **force** – Overwrite output file, if force is True
- **dry_run** – Return GATK command, if its true, default False
- **gatk_param_list** – List of additional params for BQSR, default None

Returns
GATK commandline

run_HaplotypeCaller(*input_bam*, *output_vcf_path*, *dbsnp_vcf*, *emit_gvcf=True*, *force=False*, *dry_run=False*, *gatk_param_list=None*)

A method for running GATK HaplotypeCaller

Parameters

- **input_bam** – A input bam file
- **output_vcf_path** – A output vcf filepath
- **dbsnp_vcf** – A dbsnp vcf file
- **emit_gvcf** – A toggle for GVCF generation, default True
- **force** – Overwrite output file, if force is True
- **dry_run** – Return GATK command, if its true, default False
- **gatk_param_list** – List of additional params for BQSR, default None

Returns
GATK commandline

RSEM utils

class igf_data.utils.tools.rsem_utils.**RSEM_utils**(*rsem_exe_dir*, *reference_rsem*, *input_bam*, *threads=1*, *memory_limit=4000*, *use_ephemeral_space=0*)

A python wrapper for running RSEM tool

Parameters

- **rsem_exe_dir** – RSEM executable path
- **reference_rsem** – RSEM reference transcriptome path
- **input_bam** – Input bam file path for RSEM
- **threads** – No. of threads for RSEM run, default 1
- **memory_limit** – Memory usage limit for RSEM, default 4Gb
- **use_ephemeral_space** – A toggle for temp dir settings, default 0

run_rsem_calculate_expression(*output_dir*, *output_prefix*, *paired_end=True*, *strandedness='reverse'*, *options=None*, *force=True*)

A method for running RSEM rsem-calculate-expression tool from alignment file

Parameters

- **output_dir** – A output dir path
- **output_prefix** – A output file prefix
- **paired_end** – A toggle for paired end data, default True
- **strandedness** – RNA strand information, default reverse for Illumina TruSeq allowed values are none, forward and reverse
- **options** – A dictionary for rsem run, default None
- **force** – Overwrite existing data if force is True, default False

Returns

RSEM commandline, output file list and logfile

Samtools utils

igf_data.utils.tools.samtools_utils.**convert_bam_to_cram**(*samtools_exe*, *bam_file*, *reference_file*, *cram_path*, *threads=1*, *singularity_image=None*, *force=False*, *dry_run=False*, *use_ephemeral_space=0*)

A function for converting bam files to cram using pysam utility

Parameters

- **samtools_exe** – samtools executable path
- **bam_file** – A bam filepath with / without index. Index file will be created if its missing
- **reference_file** – Reference genome fasta filepath
- **cram_path** – A cram output file path
- **threads** – Number of threads to use for conversion, default 1
- **force** – Output cram will be overwritten if force is True, default False
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

- **use_ephemeral_space** – A toggle for temp dir settings, default 0
- **singularity_image** – Singularity image path, default None

Returns
None

Raises

- **IOError** – It raises IOError if no input or reference fasta file found or output file already present and force is not True
- **ValueError** – It raises ValueError if bam_file doesn't have .bam extension or cram_path doesn't have .cram extension

`igf_data.utils.tools.samtools_utils.`**filter_bam_file**(*samtools_exe*, *input_bam*, *output_bam*, *samFlagInclude=None*, *reference_file=None*, *samFlagExclude=None*, *threads=1*, *mapq_threshold=20*, *cram_out=False*, *singularity_image=None*, *index_output=True*, *dry_run=False*)

A function for filtering bam file using samtools view

Parameters

- **samtools_exe** – Samtools path
- **input_bam** – Input bamfile path
- **output_bam** – Output bamfile path
- **samFlagInclude** – Sam flags to keep, default None
- **reference_file** – Reference genome fasta filepath
- **samFlagExclude** – Sam flags to exclude, default None
- **threads** – Number of threads to use, default 1
- **mapq_threshold** – Skip alignments with MAPQ smaller than this value, default None
- **index_output** – Index output bam, default True
- **cram_out** – Output cram file, default False
- **singularity_image** – Singularity image path, default None
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
Samtools command

`igf_data.utils.tools.samtools_utils.`**index_bam_or_cram**(*samtools_exe*, *input_path*, *threads=1*, *singularity_image=None*, *dry_run=False*)

A method for running samtools index

Parameters

- **samtools_exe** – samtools executable path
- **input_path** – Alignment filepath
- **singularity_image** – Singularity image path, default None
- **threads** – Number of threads to use for conversion, default 1
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
samtools cmd list

`igf_data.utils.tools.samtools_utils.`**`merge_multiple_bam`**(*samtools_exe*, *input_bam_list*, *output_bam_path*, *sorted_by_name=False*, *singularity_image=None*, *use_ephemeral_space=0*, *threads=1*, *force=False*, *dry_run=False*, *index_output=True*)

A function for merging multiple input bams to a single output bam

Parameters

- **`samtools_exe`** – samtools executable path
- **`input_bam_list`** – A file containing list of bam filepath
- **`output_bam_path`** – A bam output filepath
- **`sorted_by_name`** – Sort bam file by read_name, default False (for coordinate sorted bams)
- **`threads`** – Number of threads to use for merging, default 1
- **`force`** – Output bam file will be overwritten if force is True, default False
- **`index_output`** – Index output bam, default True
- **`singularity_image`** – Singularity image path, efault None
- **`use_ephemeral_space`** – A toggle for temp dir settings, default 0
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False

Returns
samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_flagstat`**(*samtools_exe*, *bam_file*, *output_dir*, *threads=1*, *force=False*, *singularity_image=None*, *output_prefix=None*, *dry_run=False*)

A method for generating bam flagstat output

Parameters

- **`samtools_exe`** – samtools executable path
- **`bam_file`** – A bam filepath with / without index. Index file will be created if its missing
- **`output_dir`** – Bam flagstat output directory path
- **`output_prefix`** – Output file prefix, default None
- **`threads`** – Number of threads to use for conversion, default 1
- **`force`** – Output flagstat file will be overwritten if force is True, default False
- **`singularity_image`** – Singularity image path, default None
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False

Returns
Output file path and a list containing samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_idxstat`**(*samtools_exe*, *bam_file*, *output_dir*, *output_prefix=None*, *singularity_image=None*, *force=False*, *dry_run=False*)

A function for running samtools index stats generation

Parameters

- **`samtools_exe`** – samtools executable path
- **`bam_file`** – A bam filepath with / without index. Index file will be created if its missing
- **`output_dir`** – Bam idxstats output directory path
- **`output_prefix`** – Output file prefix, default None
- **`force`** – Output idxstats file will be overwritten if force is True, default False
- **`singularity_image`** – Singularity image path, default None
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False

Returns

Output file path and a list containing samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_stats`**(*samtools_exe*, *bam_file*, *output_dir*, *threads=1*, *force=False*, *singularity_image=None*, *output_prefix=None*, *dry_run=False*)

A method for generating samtools stats output

Parameters

- **`samtools_exe`** – samtools executable path
- **`bam_file`** – A bam filepath with / without index. Index file will be created if its missing
- **`output_dir`** – Bam stats output directory path
- **`output_prefix`** – Output file prefix, default None
- **`threads`** – Number of threads to use for conversion, default 1
- **`singularity_image`** – Singularity image path, default None
- **`force`** – Output flagstat file will be overwritten if force is True, default False
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False

Returns

Output file path, list containing samtools command and a list containing the SN matrics of report

`igf_data.utils.tools.samtools_utils.`**`run_samtools_view`**(*samtools_exe*, *input_file*, *output_file*, *reference_file=None*, *force=True*, *cram_out=False*, *threads=1*, *samtools_params=None*, *singularity_image=None*, *index_output=True*, *dry_run=False*, *use_ephemeral_space=0*)

A function for running samtools view command

Parameters

- **`samtools_exe`** – samtools executable path
- **`input_file`** – An input bam filepath with / without index. Index file will be created if its missing

- **output_file** – An output file path
- **reference_file** – Reference genome fasta filepath, default None
- **force** – Output file will be overwritten if force is True, default True
- **threads** – Number of threads to use for conversion, default 1
- **samtools_params** – List of samtools param, default None
- **index_output** – Index output file, default True
- **singularity_image** – Singularity image path, default None
- **dry_run** – A toggle for returning the samtools command without actually running it, default False
- **use_ephemeral_space** – A toggle for temp dir settings, default 0

Returns
Samtools command as list

igf_data.utils.tools.samtools_utils.**run_sort_bam**(*samtools_exe*, *input_bam_path*, *output_bam_path*, *sort_by_name=False*, *use_ephemeral_space=0*, *singularity_image=None*, *threads=1*, *force=False*, *dry_run=False*, *cram_out=False*, *index_output=True*, *sort_params=None*)

A function for sorting input bam file and generate a output bam

Parameters

- **samtools_exe** – samtools executable path
- **input_bam_path** – A bam filepath
- **output_bam_path** – A bam output filepath
- **sort_by_name** – Sort bam file by read_name, default False (for coordinate sorting)
- **threads** – Number of threads to use for sorting, default 1
- **force** – Output bam file will be overwritten if force is True, default False
- **cram_out** – Output cram file, default False
- **index_output** – Index output bam, default True
- **singularity_image** – Singularity image path, default None
- **use_ephemeral_space** – A toggle for temp dir settings, default 0
- **dry_run** – A toggle for returning the samtools command without actually running it, default False
- **sort_params** – A list of params for samtools sort, default None

Returns
None

STAR utils

class `igf_data.utils.tools.star_utils.`**`Star_utils`**(*star_exe*, *input_files*, *genome_dir*, *reference_gtf*, *output_dir*, *output_prefix*, *threads=1*, *use_ephemeral_space=0*)

A wrapper python class for running STAR alignment

Parameters

- **`star_exe`** – STAR executable path
- **`input_files`** – List of input files for running alignment
- **`genome_dir`** – STAR reference transcriptome path
- **`reference_gtf`** – Reference GTF file for gene annotation
- **`output_dir`** – Path for output alignment and results
- **`output_prefix`** – File output prefix
- **`threads`** – No. of threads for STAR run, default 1
- **`use_ephemeral_space`** – A toggle for temp dir settings, default 0

`generate_aligned_bams`(*two_pass_mode=True*, *dry_run=False*, *star_patameters=('--outFilterMultimapNmax', 20, '--alignSJoverhangMin', 8, '--alignSJDBoverhangMin', 1, '--outFilterMismatchNmax', 999, '--outFilterMismatchNoverReadLmax', 0.04, '--alignIntronMin', 20, '--alignIntronMax', 1000000, '--alignMatesGapMax', 1000000, '--limitBAMsortRAM', 12000000000)*)

A method running star alignment

Parameters

- **`two_pass_mode`** – Run two-pass mode of star, default True
- **`dry_run`** – A toggle forreturning the star cmd without actual run, default False
- **`star_patameters`** – A dictionary of star parameters, default encode parameters

Returns
A genomic_bam and a transcriptomic bam,log file, gene count file and star commandline

`generate_rna_bigwig`(*bedGraphToBigWig_path*, *chrom_length_file*, *bedsort_path*, *stranded=True*, *dry_run=False*)

A method for generating bigWig signal tracks from star aligned bams files

Parameters

- **`bedGraphToBigWig_path`** – bedGraphToBigWig executable path
- **`chrom_length_file`** – A file containing chromosome length, e.g. .fai file
- **`bedsort_path`** – bedSort executable path

:param stranded:Param for stranded analysis, default True :param dry_run: A toggle forreturning the star cmd without actual run, default False :returns: A list of bigWig files and star commandline

Subread utils

`igf_data.utils.tools.subread_utils.`**`run_featureCounts`**(*featurecounts_exe*, *input_gtf*, *input_bams*, *output_file*, *thread=1*, *use_ephemeral_space=0*, *options=None*)

A wrapper method for running featureCounts tool from subread package

Parameters

- **`featurecounts_exe`** – Path of featureCounts executable
- **`input_gtf`** – Input gtf file path
- **`input_bams`** – input bam files
- **`output_file`** – Output filepath
- **`thread`** – Thread counts, default is 1
- **`options`** – FeaturCcount options, default in None
- **`use_ephemeral_space`** – A toggle for temp dir settings, default 0

Returns
A summary file path and featureCounts command

Reference genome fetch utils

`class` `igf_data.utils.tools.reference_genome_utils.`**`Reference_genome_utils`**(*genome_tag*, *dbsession_class*, *genome_fasta_type='GENOME_FA*, *fasta_fai_type='GENOME_FAI'*, *genome_dict_type='GENOME_DIC*, *gene_gtf_type='GENE_GTF'*, *gene_reflat_type='GENE_REFFLA*, *gene_rsem_type='TRANSCRIPTOM*, *bwa_ref_type='GENOME_BWA'*, *min-imap2_ref_type='GENOME_MINI*, *bowtie2_ref_type='GENOME_BOW*, *tenx_ref_type='TRANSCRIPTOME*, *star_ref_type='TRANSCRIPTOME*, *genome_dbsnp_type='DBSNP_VCI*, *gatk_snp_ref_type='GATK_SNP_R*, *gatk_indel_ref_type='INDEL_LIST*, *riboso-mal_interval_type='RIBOSOMAL_*, *black-list_interval_type='BLACKLIST_BI*, *genome_twobit_uri_type='GENOM*

A class for accessing different components of the reference genome for a specific build

`get_blacklist_region_bed`(*check_missing=False*)

A method for fetching blacklist interval filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_dbsnp_vcf`(*check_missing=True*)

A method for fetching filepath for dbSNP vcf file, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_gatk_indel_ref`(*check_missing=True*)

A method for fetching filepaths for INDEL files from GATK bundle, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A list of filepaths

`get_gatk_snp_ref`(*check_missing=True*)

A method for fetching filepaths for SNP files from GATK bundle, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A list of filepaths

`get_gene_gtf`(*check_missing=True*)

A method for fetching reference gene annotation gtf filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_gene_reflat`(*check_missing=True*)

A method for fetching reference gene annotation refflat filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_generic_ref_files`(*collection_type*, *check_missing=True*)

A method for fetching filepath for generic reference genome file, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string or list (if more than one found)

`get_genome_bowtie2`(*check_missing=True*)

A method for fetching filepath of Bowtie2 reference index, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_genome_bwa`(*check_missing=True*)

A method for fetching filepath of BWA reference index, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_genome_dict`(*check_missing=True*)
A method for fetching reference genome dictionary filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_genome_fasta`(*check_missing=True*)
A method for fetching reference genome fasta filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_genome_fasta_fai`(*check_missing=True*)
A method for fetching reference genome fasta fai index filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_genome_minimap2`(*check_missing=True*)
A method for fetching filepath of Minimap2 reference index, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_ribosomal_interval`(*check_missing=True*)
A method for fetching ribosomal interval filepath for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_transcriptome_rsem`(*check_missing=False*)
A method for fetching filepath of RSEM reference transcriptome, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

`get_transcriptome_star`(*check_missing=False*)
A method for fetching filepath of STAR reference transcriptome, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

get_transcriptome_tenx(*check_missing=True*)

A method for fetching filepath of 10X Cellranger reference transcriptome, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A filepath string

get_twobit_genome_url(*check_missing=True*)

A method for fetching filepath for twobit genome url, for a specific genome build

Parameters
`check_missing` – A toggle for checking errors for missing files, default True

Returns
A url string

Samtools utils

igf_data.utils.tools.samtools_utils.**convert_bam_to_cram**(*samtools_exe*, *bam_file*, *reference_file*, *cram_path*, *threads=1*, *singularity_image=None*, *force=False*, *dry_run=False*, *use_ephemeral_space=0*)

A function for converting bam files to cram using pysam utility

Parameters

- `samtools_exe` – samtools executable path
- `bam_file` – A bam filepath with / without index. Index file will be created if its missing
- `reference_file` – Reference genome fasta filepath
- `cram_path` – A cram output file path
- `threads` – Number of threads to use for conversion, default 1
- `force` – Output cram will be overwritten if force is True, default False
- `dry_run` – A toggle for returning the samtools command without actually running it, default False
- `use_ephemeral_space` – A toggle for temp dir settings, default 0
- `singularity_image` – Singularity image path, default None

Returns
None

Raises

- `IOError` – It raises IOError if no input or reference fasta file found or output file already present and force is not True
- `ValueError` – It raises ValueError if bam_file doesn't have .bam extension or cram_path doesn't have .cram extension

igf_data.utils.tools.samtools_utils.**filter_bam_file**(*samtools_exe*, *input_bam*, *output_bam*, *samFlagInclude=None*, *reference_file=None*, *samFlagExclude=None*, *threads=1*, *mapq_threshold=20*, *cram_out=False*, *singularity_image=None*, *index_output=True*, *dry_run=False*)

A function for filtering bam file using samtools view

Parameters

- **samtools_exe** – Samtools path
- **input_bam** – Input bamfile path
- **output_bam** – Output bamfile path
- **samFlagInclude** – Sam flags to keep, default None
- **reference_file** – Reference genome fasta filepath
- **samFlagExclude** – Sam flags to exclude, default None
- **threads** – Number of threads to use, default 1
- **mapq_threshold** – Skip alignments with MAPQ smaller than this value, default None
- **index_output** – Index output bam, default True
- **cram_out** – Output cram file, default False
- **singularity_image** – Singularity image path, default None
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
Samtools command

igf_data.utils.tools.samtools_utils.**index_bam_or_cram**(*samtools_exe*, *input_path*, *threads=1*, *singularity_image=None*, *dry_run=False*)

A method for running samtools index

Parameters

- **samtools_exe** – samtools executable path
- **input_path** – Alignment filepath
- **singularity_image** – Singularity image path, default None
- **threads** – Number of threads to use for conversion, default 1
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
samtools cmd list

igf_data.utils.tools.samtools_utils.**merge_multiple_bam**(*samtools_exe*, *input_bam_list*, *output_bam_path*, *sorted_by_name=False*, *singularity_image=None*, *use_ephemeral_space=0*, *threads=1*, *force=False*, *dry_run=False*, *index_output=True*)

A function for merging multiple input bams to a single output bam

Parameters

- **samtools_exe** – samtools executable path
- **input_bam_list** – A file containing list of bam filepath
- **output_bam_path** – A bam output filepath

- **sorted_by_name** – Sort bam file by read_name, default False (for coordinate sorted bams)
- **threads** – Number of threads to use for merging, default 1
- **force** – Output bam file will be overwritten if force is True, default False
- **index_output** – Index output bam, default True
- **singularity_image** – Singularity image path, efault None
- **use_ephemeral_space** – A toggle for temp dir settings, default 0
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_flagstat`**(*samtools_exe*, *bam_file*, *output_dir*, *threads=1*, *force=False*, *singularity_image=None*, *output_prefix=None*, *dry_run=False*)

A method for generating bam flagstat output

Parameters

- **samtools_exe** – samtools executable path
- **bam_file** – A bam filepath with / without index. Index file will be created if its missing
- **output_dir** – Bam flagstat output directory path
- **output_prefix** – Output file prefix, default None
- **threads** – Number of threads to use for conversion, default 1
- **force** – Output flagstat file will be overwritten if force is True, default False
- **singularity_image** – Singularity image path, default None
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
Output file path and a list containing samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_idxstat`**(*samtools_exe*, *bam_file*, *output_dir*, *output_prefix=None*, *singularity_image=None*, *force=False*, *dry_run=False*)

A function for running samtools index stats generation

Parameters

- **samtools_exe** – samtools executable path
- **bam_file** – A bam filepath with / without index. Index file will be created if its missing
- **output_dir** – Bam idxstats output directory path
- **output_prefix** – Output file prefix, default None
- **force** – Output idxstats file will be overwritten if force is True, default False
- **singularity_image** – Singularity image path, default None
- **dry_run** – A toggle for returning the samtools command without actually running it, default False

Returns
Output file path and a list containing samtools command

`igf_data.utils.tools.samtools_utils.`**`run_bam_stats`**(*samtools_exe*, *bam_file*, *output_dir*, *threads=1*, *force=False*, *singularity_image=None*, *output_prefix=None*, *dry_run=False*)

A method for generating samtools stats output

Parameters

- **`samtools_exe`** – samtools executable path
- **`bam_file`** – A bam filepath with / without index. Index file will be created if its missing
- **`output_dir`** – Bam stats output directory path
- **`output_prefix`** – Output file prefix, default None
- **`threads`** – Number of threads to use for conversion, default 1
- **`singularity_image`** – Singularity image path, default None
- **`force`** – Output flagstat file will be overwritten if force is True, default False
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False

Returns
Output file path, list containing samtools command and a list containing the SN matrics of report

`igf_data.utils.tools.samtools_utils.`**`run_samtools_view`**(*samtools_exe*, *input_file*, *output_file*, *reference_file=None*, *force=True*, *cram_out=False*, *threads=1*, *samtools_params=None*, *singularity_image=None*, *index_output=True*, *dry_run=False*, *use_ephemeral_space=0*)

A function for running samtools view command

Parameters

- **`samtools_exe`** – samtools executable path
- **`input_file`** – An input bam filepath with / without index. Index file will be created if its missing
- **`output_file`** – An output file path
- **`reference_file`** – Reference genome fasta filepath, default None
- **`force`** – Output file will be overwritten if force is True, default True
- **`threads`** – Number of threads to use for conversion, default 1
- **`samtools_params`** – List of samtools param, default None
- **`index_output`** – Index output file, default True
- **`singularity_image`** – Singularity image path, default None
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False
- **`use_ephemeral_space`** – A toggle for temp dir settings, default 0

Returns
Samtools command as list

`igf_data.utils.tools.samtools_utils.`**`run_sort_bam`**(*samtools_exe*, *input_bam_path*, *output_bam_path*, *sort_by_name=False*, *use_ephemeral_space=0*, *singularity_image=None*, *threads=1*, *force=False*, *dry_run=False*, *cram_out=False*, *index_output=True*, *sort_params=None*)

A function for sorting input bam file and generate a output bam

Parameters

- **`samtools_exe`** – samtools executable path
- **`input_bam_path`** – A bam filepath
- **`output_bam_path`** – A bam output filepath
- **`sort_by_name`** – Sort bam file by read_name, default False (for coordinate sorting)
- **`threads`** – Number of threads to use for sorting, default 1
- **`force`** – Output bam file will be overwritten if force is True, default False
- **`cram_out`** – Output cram file, default False
- **`index_output`** – Index output bam, default True
- **`singularity_image`** – Singularity image path, default None
- **`use_ephemeral_space`** – A toggle for temp dir settings, default 0
- **`dry_run`** – A toggle for returning the samtools command without actually running it, default False
- **`sort_params`** – A list of params for samtools sort, default None

Returns
None

Scanpy utils

2.2.4 Metadata processing

Register metadata for new projects

class igf_data.process.seqrun_processing.find_and_register_new_project_data.**Find_and_register_new_pr**

A class for finding new data for project and registering them to the db. Account for new users will be created in irods server and password will be mailed to them.

Parameters

- **projet_info_path** – A directory path for project info files
- **dbconfig** – A json dbconfig file
- **check_hpc_user** – Guess the hpc user name, True or False, default: False
- **hpc_user** – A hpc user name, default is None
- **hpc_address** – A hpc host address, default is None
- **ldap_server** – A ldap server address for search, default is None
- **user_account_template** – A template file for user account activation email
- **log_slack** – Enable or disable sending message to slack, default: True
- **slack_config** – A slack config json file, required if log_slack is True
- **project_lookup_column** – project data lookup column, default project_igf_id
- **user_lookup_column** – user data lookup column, default email_id
- **sample_lookup_column** – sample data lookup column, default sample_igf_id
- **data_authority_column** – data authority column name, default data_authority
- **setup_irods** – Setup irods account for user, default is True
- **notify_user** – Send email notification to user, default is True
- **default_user_email** – Add another user as the default collaborator for all new projects, default igf@imperial.ac.uk

- **barcode_check_keyword** – Project attribute name for barcode check settings, default barcode_check
- **sendmail_exe** – Sendmail executable path, default /usr/sbin/sendmail

process_project_data_and_account()

A method for finding new project info and registering them to database and user account creation

Update experiment metadata from sample attributes

class igf_data.process.metadata.experiment_metadata_updator.**Experiment_metadata_updator**(*dbconfig_file*, *log_slack=True*, *slack_config=N*

A class for updating metadata for experiment table in database

update_metadta_from_sample_attribute(*experiment_igf_id=None*, *sample_attribute_names=('library_source', 'library_strategy', 'experiment_type')*)

A method for fetching experiment metadata from sample_attribute tables :param experiment_igf_id: An experiment igf id for updating only a selected experiment, default None for all experiments :param sample_attribute_names: A list of sample attribute names to look for experiment metadata,

default: library_source, library_strategy, experiment_type

2.2.5 Sequencing run

Process samplesheet file

class igf_data.illumina.samplesheet.**SampleSheet**(*infile*, *data_header_name=('Data', 'BCLConvert_Data')*)

A class for processing SampleSheet files for Illumina sequencing runs

Parameters

- **infile** – A samplesheet file
- **data_header_name** – name of the data section, default Data

add_pseudo_lane_for_miseq(*lane='1'*)

A method for adding pseudo lane information for the nextseq platform

Parameters
lane – A lane id for pseudo lane value

add_pseudo_lane_for_nextseq(*lanes=('1', '2', '3', '4')*)

A method for adding pseudo lane information for the nextseq platform

Parameters
lanes – A list of pseudo lanes, default ['1','2','3','4']

:returns:None

check_sample_header(*section*, *condition_key*, *return_values=False*)

Function for checking SampleSheet header

Parameters

- **section** – A field name for header info check
- **condition_key** – A condition key for header info check
- **return_values** – Taggole for a list of return values instead of zero or match counts

Returns
zero if its not present or number of occurrence of the term, or list of matching items with return_values=True

`filter_sample_data`(*condition_key*, *condition_value*, *method='include'*, *lane_header='Lane'*, *lane_default_val='1'*)

Function for filtering SampleSheet data based on matching condition

Parameters

- `condition_key` – A samplesheet column name
- `condition_value` – A keyword present in the selected column
- `method` – 'include' or 'exclude' for adding or removing selected column from the samplesheet default is include

`get_index_count`()

A function for getting index length counts

Returns
A dictionary, with the index columns as the key

`get_indexes`()

A method for retrieving the indexes from the samplesheet

Returns
A list of index barcodes

`get_lane_count`(*lane_field='Lane'*, *target_platforms=('HiSeq', 'NovaSeq')*)

Function for getting the lane information for HiSeq runs It will return 1 for both MiSeq and NextSeq runs

Parameters

- `lane_field` – Column name for lane info, default 'Lane'
- `target_platform` – Hiseq platform tag, default 'HiSeq'

Returns
A list of lanes present in samplesheet file

`get_platform_name`(*section='Header'*)

Function for getting platform details from samplesheet header

Parameters
`section` – File section for lookup, default 'Header'

`get_project_and_lane`(*project_tag='Sample_Project'*, *lane_tag='Lane'*)

A method for fetching project and lane information from samplesheet

Parameters

- `project_tag` – A string for project name column in the samplesheet, default Sample_Project
- `lane_tag` – A string for Lane id column in the samplesheet, default Lane

Returns
A list of project name (for all) and lane information (only for hiseq)

`get_project_names`(*tag='sample_project'*)

Function for retrieving unique project names from samplesheet. If there are multiple matching headers, the first column will be used

Parameters
`tag` – Name of tag for project lookup, default sample_project

Returns
A list of unique project name

`get_reverse_complement_index`(*index_field='index2'*)

A function for changing the I5_index present in the index2 field of the samplesheet to intsreverse complement base

Parameters
`index_field` – Column name for index 2, default index2

`group_data_by_index_length`()

Function for grouping samplesheet rows based on the combined length of index columns By default, this function removes Ns from the index

Returns
A dictionary of samplesheet objects, with combined index length as the key

`modify_sample_header`(*section*, *type*, *condition_key*, *condition_value=''*)

Function for modifying SampleSheet header

Parameters

- `section` – A field name for header info check
- `condition_key` – A condition key for header info check
- `type` – Mode type, 'add' or 'remove'
- `condition_value` – Its is required for 'add' type

`print_sampleSheet`(*outfile*)

Function for printing output SampleSheet

Parameters
`outfile` – A output samplesheet path

`validate_samplesheet_data`(*schema_json*)

A method for validation of samplesheet data

Parameters
`schema` – A JSON schema for validation of the samplesheet data

:return a list of error messages or an empty list if no error found

Fetch read cycle info from RunInfo.xml file

class `igf_data.illumina.runinfo_xml.RunInfo_xml`(*xml_file*)

A class for reading runinfo xml file from illumina sequencing runs

Parameters
`xml_file` – A runinfo xml file

`get_flowcell_name`()

A method for accessing flowcell name from the runinfo xml file

`get_platform_number`()

Function for fetching the instrument series number

`get_reads_stats`(*root_tag='read'*, *number_tag='number'*, *tags=('isindexedread', 'numcycles', 'isreversecomplement')*)

A method for getting read and index stats from the RunInfo.xml file

Parameters

- `root_tag` – Root tag for xml file, default read

- **number_tag** – Number tag for xml file, default number
- **tags** – List of tags for xml lookup, default [‘isindexedread’, ‘numcycles’, ‘isreversecomplement’]

Returns
A dictionary with the read number as the key

Fetch flowcell info from runparameters xml file

class igf_data.illumina.runparameters_xml.**RunParameter_xml**(*xml_file*)

A class for reading runparameters xml file from Illumina sequencing runs

Parameters
xml_file – A runparameters xml file

get_hiseq_flowcell()

A method for fetching flowcell details for hiseq run

Returns
Flowcell info or None (for MiSeq, NextSeq or NovaSeq runs)

Find and process new sequencing run for demultiplexing

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**calculate_file_md5**(*seqrun_info*, *md5_out*, *seqrun_path*, *file_suffix='md5.json'*, *exclude_dir=()*)

A method for file md5 calculation for all the sequencing run files

Parameters

- **seqrun_info** – A dictionary containing sequencing run information
- **md5_out** – JSON md5 file output directory
- **file_suffix** – Suffix information for new JSON md5 files, default: md5.json
- **exclude_dir** – A list of directories to exclude from the file look up

Returns

Output is a dictionary of json files

{seqrun_name: seqrun_md5_list_path} Format of the json file [{“seqrun_file_name”:”file_path”,

”file_md5”:”md5_value”, “file_size”:”size”}]

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**check_finished_seqrun_dir**(*seqrun_dir*, *seqrun_path*, *required_files=(*
'SampleSheet.csv', *'RunInfo.xml')*)

A method for checking complete sequencing run directory

Parameters

- **seqrun_dir** – A list of sequencing run names
- **seqrun_path** – A directory path for new sequencing run look up
- **required_files** – A list of files to check before marking sequencing run as complete, default: 'RTAComplete.txt','SampleSheet.csv','RunInfo.xml'

Returns
A dictionary containing valid sequencing run information

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**check_for_registered_project_and_samp**

A method for fetching project and sample records from samplesheet and checking for registered samples in db

Parameters

- **seqrun_info** – A dictionary containing seqrun name and path as key and values
- **dbconfig** – A database configuration file
- **samplesheet_file** – Name of samplesheet file, default is SampleSheet.csv

Returns
A dictionary containing the new run information A string message containing database checking information

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**check_seqrun_dir_in_db**(*all_seqrun_dir*, *db-con-fig*)

A method for checking existing seqrun dirs in database

Parameters

- **all_seqrun_dir** – list of seqrun dirs to check
- **dbconfig** – dbconfig

Returns
A list containing new sequencing run information

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**find_new_seqrun_dir**(*path*, *db-con-fig*)

A method for check and finding new sequencing run directory

Parameters

- **path** – A directory path for new sequencing run lookup
- **dbconfig** – A database configuration file

Returns
A list of new sequencing run names for processing

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**load_seqrun_files_to_db**(*seqrun_info*, *se-qrun_md5_info*, *db-con-fig*, *file_type='ILLUM*

A method for loading md5 lists to collection and files table

Parameters

- **seqrun_info** – A dictionary containing the sequencing run information
- **seqrun_md5_info** – A dictionary containing the sequencing run JSON md5 file info
- **dbconfig** – A database configuration file
- **file_type** – A collection type information for loading the JSON files to database

Returns
Nill

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**prepare_seqrun_for_db**(*seqrun_name, seqrun_path, session_class*)

A method for preparing seqrun data for database

Parameters

- **seqrun_name** – A sequencing run name
- **seqrun_path** – A directory path for sequencing run look up
- **session_class** – A database session class

Returns
A dictionary containing information to populate the seqrun table in database

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**seed_pipeline_table_for_new_seqrun**(*pip db con fig*

A method for seeding pipelines for the new seqruns

Parameters

- **pipeline_name** – A pipeline name
- **dbconfig** – A dbconfig file

Returns
Nill

igf_data.process.seqrun_processing.find_and_process_new_seqrun.**validate_samplesheet_for_seqrun**(*seqrun schema out-put_di sam-pleshe*

A method for validating samplesheet and writing errors to a report file

Parameters

- **seqrun_info** – A dictionary containing seqrun name and path as key and values
- **schema_json** – A json schema for samplesheet validation
- **output_dir** – A directory path for writing output report files
- **samplesheet_file** – Samplesheet filename, default 'SampleSheet.csv'

Returns
new_seqrun_info, A new dictionary containing seqrun name and path as key and values

Returns
error_file_list, A dictionary containing seqrun name and error file paths as key and values

2.2.6 Demultiplexing

Bases mask calculation

`class igf_data.illumina.basesMask.`**`BasesMask`**(*samplesheet_file*, *runinfo_file*, *read_offset*, *index_offset*)

A class for bases mask value calculation for demultiplexing of sequencing runs

Parameters

- **`samplesheet_file`** – A samplesheet file containing sample index barcodes
- **`runinfo_file`** – A runinfo xml file from sequencing run
- **`read_offset`** – Read offset value in bp
- **`index_offset`** – Index offset value in bp

`calculate_bases_mask`(*numcycle_label='numcycles'*, *isindexedread_label='isindexedread'*)

A method for bases mask value calculation

Parameters

- **`numcycle_label`** – Cycle label in runinfo xml file, default numcycles
- **`isindexedread_label`** – Index cycle label in runinfo xml file, default isindexedread

Returns
A formatted bases mask value for bcl2fastq run

Copy bcl files for demultiplexing

`class igf_data.process.moveBclFilesForDemultiplexing.`**`moveBclTilesForDemultiplexing`**(*input_dir*, *output_dir*, *samplesheet*, *run_info_xml*, *force=False*, *platform_model=None*, *tiles_list=(1101,)*)

A class for copying BCL files for a list of tiles to a specific dir

Parameters

- **`input_dir`** – Input run dir
- **`output_dir`** – Target output dir
- **`samplesheet`** – Samplesheet filepath
- **`run_info_xml`** – RunInfo.xml file path
- **`platform_model`** – Platform model, default None
- **`force`** – Force copy existing file, default False
- **`tiles_list`** – List of times to copy, default (1101,)

  obj = moveBclTilesForDemultiplexing(**kwargs) obj.copy_bcl_files()

copy_bcl_files()

A function for transferring bcl files for selected tiles to a target dir

Collect demultiplexed fastq files to database

class igf_data.process.seqrun_processing.collect_seqrun_fastq_to_db.**Collect_seqrun_fastq_to_db**(*fastq_d model_ se- qrun_i ses- sion_c flow- cell_id sam- pleshe sam- pleshe col- lec- tion_ty file_loc col- lec- tion_ta man- i- fest_na sin- gle- cell_ta*

A class for collecting raw fastq files after demultiplexing and storing them in database. Additionally this will also create relevant entries for the experiment and run tables in database

Parameters

- **fastq_dir** – A directory path for file look up
- **model_name** – Sequencing platform information
- **seqrun_igf_id** – Sequencing run name
- **session_class** – A database session class
- **flowcell_id** – Flowcell information for the run
- **samplesheet_file** – Samplesheet filepath
- **samplesheet_filename** – Name of the samplesheet file, default SampleSheet.csv
- **collection_type** – Collection type information for new fastq files, default demultiplexed_fastq
- **file_location** – Fastq file location information, default HPC_PROJECT
- **collection_table** – Collection table information for fastq files, default run
- **manifest_name** – Name of the file manifest file, default file_manifest.csv
- **singlecell_tag** – Samplesheet description for singlecell samples, default 10X

find_fastq_and_build_db_collection()

A method for finding fastq files and samplesheet under a run directory and loading the new files to db with their experiment and run information

It calculates following entries

- **library_name**
  Same as sample_id unless mentioned in 'Description' field of samplesheet
- **experiment_igf_id**
  library_name combined with the platform name same library sequenced in different platform will be added as separate experiemnt
- **run_igf_id**
  experiment_igf_id combined with sequencing flowcell_id and lane_id collection name: Same as run_igf_id, fastq files will be added to db collection using this id
- **collection type**
  Default type for fastq file collections are 'demultiplexed_fastq'
- **file_location**
  Default value is 'HPC_PROJECT'

Check demultiplexing barcode stats

2.2.7 Pipeline control

Reset pipeline seeds for re-processing

class igf_data.process.pipeline.modify_pipeline_seed.**Modify_pipeline_seed**(*igf_id_list*, *table_name*, *pipeline_name*, *dbconfig_file*, *log_slack=True*, *log_asana=True*, *slack_config=None*, *asana_project_id=None*, *asana_config=None*, *clean_up=True*)

A class for changing pipeline run status in the pipeline_seed table

reset_pipeline_seed_for_rerun(*seeded_label='SEEDED'*, *restricted_status_list=('SEEDED', 'RUNNING')*)

A method for setting the pipeline for re-run if the first run has failed or aborted This method will set the pipeline_seed.status as 'SEEDED' only if its not already 'SEEDED' or 'RUNNING' :param seeded_label: A text label for seeded status, default SEEDED :param restricted_status_list: A list of pipeline status to exclude from the search,

default ['SEEDED','RUNNING']

Reset samplesheet files after modification for rerunning pipeline

class `igf_data.process.seqrun_processing.reset_samplesheet_md5.`**`Reset_samplesheet_md5`**(*seqrun_path, se- qrun_igf_list, db- con- fig_file, clean_up=True, json_collection_typ log_slack=True, log_asana=True, slack_config=None, asana_project_id=1 asana_config=None sam- plesheet_name='Sa*

A class for modifying samplesheet md5 for seqrun data processing

run()

A method for resetting md5 values in the samplesheet json files for all seqrun ids

2.2.8 Demultiplexing of single cell sample

Modify samplesheet for singlecell samples

class igf_data.process.singlecell_seqrun.processsinglecellsamplesheet.ProcessSingleCellDualIndexSamp

A class for processing singlecell dual indeices on the samplesheet

Parameters

- `samplesheet_file` – A samplesheet file containg the single cell sample
- `singlecell_dual_index_barcode_json` – A json file containing 10x 3.1 dual index barcodes
- `platform` – Sequencing platform name
- `singlecell_tag` – A text keyword for the single cell sample Description, default '10X'
- `index_column` – Column name for index lookup, default 'index'
- `index2_column` – Column name for index2 lookup, default 'index2'
- `sc_barcode_index1_tag` – Index I7 tag in the json barcode file, default 'index(i7)'
- `sample_description_column` – Sample description column name in samplesheet, default 'Description'
- `index2_rule` – Rule for changing index2 barcode, default None
- `workflow_group` – A dictionary containing the I5 index tag for different platforms, default (('HISEQ4000', 'index2_workflow_b(i5)'), ('NEXTSEQ', 'index2_workflow_b(i5)'), ('NOVASEQ6000', 'index2_workflow_a(i5)'), ('NEXTSEQ2000', 'index2_workflow_a(i5)'), ('MISEQ', 'index2_workflow_a(i5)'))

modify_samplesheet_for_sc_dual_barcode(*output_samplesheet: str*, *remove_adapters: bool = True*, *adapter_trim_section: str = 'Settings'*, *adapter1_label: str = 'Adapter'*, *adapter2_label: str = 'AdapterRead2'*) → None

A method for modifying samplesheet file sor sc dual index barcodes

Parameters

- `output_samplesheet` – A file path for output samplesheet file. This file shouldn't be present.
- `remove_adapters` – Remove adapter config from samplesheet, default True
- `adapter_trim_section` – Adapter trim section name on samplesheet, default 'Settings'
- `adapter1_label` – Adapter 1 label, default 'Adapter'
- `adapter2_label` – Adapter 2 label, default 'AdapterRead2'

class igf_data.process.singlecell_seqrun.processsinglecellsamplesheet.**ProcessSingleCellSamplesheet**(*sa
su
g
ce
su
g
ce
in
d
sa
p
sa
p
o
o
sa
p
o*

A class for processing samplesheet containing single cell (10X) index barcodes It requires a json format file listing all the single cell barcodes downloaded from this page https://support.10xgenomics.com/single-cell-gene-expression/sequencing/doc/ specifications-sample-index-sets-for-single-cell-3

Parameters

- **samplesheet_file** – A samplesheet containing single cell samples
- **singlecell_barcode_json** – A JSON file listing single cell indexes
- **singlecell_tag** – A text keyword for the single cell sample description
- **index_column** – Column name for index lookup, default 'index'
- **sample_description_column** – Sample description column name in samplesheet, default 'Description'
- **sample_id_column** – Column name for sample_id lookup, default 'Sample_ID'
- **sample_name_column** – Column name for sample_name lookup, default 'Sample_NAme'
- **orig_sample_id** – Column name for keeping original sample ids, default 'Original_Sample_ID'
- **orig_sample_name** – Column name for keeping original sample_names, default: 'Original_Sample_Name'
- **orig_index** – Column name for keeping original index, default 'Original_index'

change_singlecell_barcodes(*output_samplesheet*)

A method for replacing single cell index codes present in the samplesheet with the four index sequences. This method will create 4 samplesheet entries for each of the single cell samples with _1 to _4 suffix and relevant indexes

Parameters
output_samplesheet – A file name of the output samplesheet

Merge fastq files for single cell samples

class igf_data.process.singlecell_seqrun.mergesinglecellfastq.**MergeSingleCellFastq**(*fastq_dir: str, samplesheet: str, platform_name: str, singlecell_tag: str = '10X', sampleid_col: str = 'Sample_ID', samplename_col: str = 'Sample_Name', use_ephemeral_space. bool = False, orig_sampleid_col: str = 'Original_Sample_ID', description_col: str = 'Description', orig_samplename_col str = 'Original_Sample_Name', project_col: str = 'Sample_Project', lane_col: str = 'Lane', pseudo_lane_list: tuple = ('1'*

A class for merging single cell fastq files per lane per sample

Parameters

- `fastq_dir` – A directory path containing fastq files
- `samplesheet` – A samplesheet file used demultiplexing of bcl files
- `platform_name` – A sequencing platform name
- `singlecell_tag` – A single cell keyword for description field, default '10X'
- `sampleid_col` – A keyword for sample id column of samplesheet, default 'Sample_ID'
- `samplename_col` – A keyword for sample name column of samplesheet, default 'Sample_Name'
- `orig_sampleid_col` – A keyword for original sample id column, default 'Original_Sample_ID'
- `orig_samplename_col` – A keyword for original sample name column, default 'Original_Sample_Name'
- `description_col` – A keyword for description column, default 'Description'
- `project_col` – A keyword for project column, default 'Sample_Project'
- `pseudo_lane_col` – A keyword for pseudo lane column, default 'PseudoLane'
- `lane_col` – A keyword for lane column, default 'Lane'
- `force_overwrite` – A toggle for overwriting output fastqs, default True
- `pseudo_lane_list` – Pseudo lanes for NextSeq, default ('1', '2', '3', '4')
- `use_bclconvert_settings` – Use bclconvert settings, default False

Patam use_sample_id_as_fastq_prefix
Use Sample id as fastq name, default False for Sample name

SampleSheet file should contain following columns:

- Sample_ID: A single cell sample id in the following format, SampleId_{digit}
- Sample_Name: A single cell sample name in the following format, SampleName_{digit}
- Original_Sample_ID: An IGF sample id
- Original_Sample_Name: A sample name provided by user
- Description: A single cell label, default 10X

`merge_fastq_per_lane_per_sample()`

A method for merging single cell fastq files present in input fastq_dir per lane per sample basis

2.2.9 Report page building

Configure Biodalliance genome browser for qc page

class igf_data.utils.config_genome_browser.**Config_genome_browser**(*dbsession_class, project_igf_id, collection_type_list, pipeline_name, collection_table, species_name, ref_genome_type, track_file_type=None, analysis_path_prefix='analysis', use_ephemeral_space=0, analysis_dir_structure_list=('sample_igf_id',)*)

A class for configuring genome browser input files for analysis track visualization

Parameters

- **dbsession_class** – A database session class
- **project_igf_id** – A project igf id
- **collection_type_list** – A list of collection types to include in the track
- **pipeline_name** – Name of the analysis pipeline for status checking
- **collection_table** – Name of file collection table name
- **species_name** – Species name for ref genome fetching
- **ref_genome_type** – Reference genome type for remote tracks
- **track_file_type** – Additional track file collection types
- **analysis_path_prefix** – Top level dir name for analysis files, default 'analysis'
- **use_ephemeral_space** – A toggle for temp dir settings, default 0
- **analysis_dir_structure_list** – List of keywords for sub directory paths, default ['sample_igf_id']

build_biodalliance_config(*template_file, output_file*)

A method for building biodalliance specific config file :param template_file: A template file path :param output_file: An output filepath

Process Google chart json data

igf_data.utils.gviz_utils.**convert_to_gviz_json_for_display**(*description, data, columns_order, output_file=None*)

A utility method for writing gviz format json file for data display using Google charts

:param description, A dictionary for the data table description :param data, A dictionary containing the data table :column_order, A tuple of data table column order :param output_file, Output filename, default None :returns: None if output_file name is present, or else json_data string

Generate data for QC project page

`igf_data.utils.project_data_display_utils.`**`add_seqrun_path_info`**(*input_data*, *output_file*, *seqrun_col='seqrun_igf_id'*, *flowcell_col='flowcell_id'*, *path_col='path'*)

A utility method for adding remote path to a dataframe for each sequencing runs of a project

required params: :param input_data, A input dataframe containing the following columns

seqrun_igf_id flowcell_id

:param seqrun_col, Column name for sequencing run id, default seqrun_igf_id :param flowcell_col, Column namae for flowcell id, default flowcell_id :param path_col, Column name for path, default path output_file: An output filepath for the json data

`igf_data.utils.project_data_display_utils.`**`convert_project_data_gviz_data`**(*input_data*, *sample_col='sample_igf_id'*, *read_count_col='attribute_value'*, *seqrun_col='flowcell_id'*)

A utility method for converting project's data availability information to gviz data table format https://developers.google.com/chart/interactive/docs/reference#DataTable

required params: :param input_data: A pandas data frame, it should contain following columns

sample_igf_id, flowcell_id, attribute_value (R1_READ_COUNT)

:param sample_col, Column name for sample id, default sample_igf_id :param seqrun_col, Column name for sequencing run identifier, default flowcell_id :param read_count_col, Column name for sample read counts, default attribute_value

return
a dictionary of description a list of data dictionary a tuple of column_order

Generate data for QC status page

`class igf_data.utils.project_status_utils.`**`Project_status`**(*project_igf_id: str*, *igf_session_class: Optional[Any] = None*, *dbconfig_file: Optional[Any] = None*, *seqrun_work_day: int = 2*, *analysis_work_day: int = 1*, *sequencing_resource_name: str = 'Sequencing'*, *demultiplexing_resource_name: str = 'Demultiplexing'*, *analysis_resource_name: str = 'Primary Analysis'*, *task_id_label: str = 'task_id'*, *task_name_label: str = 'task_name'*, *resource_label: str = 'resource'*, *dependencies_label: str = 'dependencies'*, *start_date_label: str = 'start_date'*, *end_date_label: str = 'end_date'*, *duration_label: str = 'duration'*, *percent_complete_label: str = 'percent_complete'*)

A class for project status fetch and gviz json file generation for Google chart grantt plot

Parameters

- **`igf_session_class`** – Database session class

- `project_igf_id` – Project igf id for database lookup
- `seqrun_work_day` – Duration for seqrun jobs in days, default 2
- `analysis_work_day` – Duration for analysis jobs in days, default 1
- `sequencing_resource_name` – Resource name for sequencing data, default Sequencing
- `demultiplexing_resource_name` – Resource name for demultiplexing data,default Demultiplexing
- `analysis_resource_name` – Resource name for analysis data, default Primary Analysis
- `task_id_label` – Label for task id field, default task_id
- `task_name_label` – Label for task name field, default task_name
- `resource_label` – Label for resource field, default resource
- `start_date_label` – Label for start date field, default start_date
- `end_date_label` – Label for end date field, default end_date
- `duration_label` – Label for duration field, default duration
- `percent_complete_label` – Label for percent complete field, default percent_complete
- `dependencies_label` – Label for dependencies field, default dependencies

`generate_gviz_json_file`(*output_file: str*, *demultiplexing_pipeline: str*, *analysis_pipeline: str*, *active_seqrun_igf_id=None*) → None

A wrapper method for writing a gviz json file with project status information

Parameters

- `output_file` – A filepath for writing project status
- `analysis_pipeline` – Name of the analysis pipeline
- `demultiplexing_pipeline` – Name of the demultiplexing pipeline
- `analysis_pipeline` – name of the analysis pipeline
- `active_seqrun_igf_id` – Igf id go the active seqrun, default None

Returns
None

`get_analysis_info`(*analysis_pipeline*)

A method for fetching all active experiments and their run status for a project

Parameters
`analysis_pipeline` – Name of the analysis pipeline

Returns
A list of dictionary containing the analysis information

`get_seqrun_info`(*active_seqrun_igf_id=None*, *demultiplexing_pipeline=None*)

A method for fetching all active sequencing runs for a project

Parameters

- `active_seqrun_igf_id` – Seqrun igf id for the current run, default None
- `demultiplexing_pipeline` – Name of the demultiplexing pipeline, default None

Returns
A dictionary containing seqrun information

`static get_status_column_order()` → list

A method for fetching column order for status json data

Returns
A list data containing the column order

`static get_status_description()` → dict

A method for getting description for status json data

Returns
A dictionary containing status info

Generate data for QC analysis page

`class igf_data.utils.project_analysis_utils.Project_analysis(`*igf_session_class, collection_type_list, remote_analysis_dir='analysis', use_ephemeral_space=0, attribute_collection_file_type='ANALYSIS_CRAM', pipeline_name='PrimaryAnalysisCombined', pipeline_seed_table='experiment', pipeline_finished_status='FINISHED', sample_id_label='SAMPLE_ID'*`)`

A class for fetching all the analysis files linked to a project

Parameters

- **`igf_session_class`** – A database session class
- **`collection_type_list`** – A list of collection type for database lookup
- **`remote_analysis_dir`** – A remote path prefix for analysis file look up, default analysis
- **`attribute_collection_file_type`** – A filetype list for fetching collection attribute records, default ('ANALYSIS_CRAM')

`get_analysis_data_for_project(`*project_igf_id, output_file, chart_json_output_file=None, csv_output_file=None, gviz_out=True, file_path_column='file_path', type_column='type', sample_igf_id_column='sample_igf_id'*`)`

A method for fetching all the analysis files for a project

Parameters

- **`project_igf_id`** – A project igf id for database lookup
- **`output_file`** – An output filepath, either a csv or a gviz json
- **`gviz_out`** – A toggle for converting output to gviz output, default is True
- **`sample_igf_id_column`** – A column name for sample igf id, default sample_igf_id
- **`file_path_column`** – A column name for file path, default file_path
- **`type_column`** – A column name for collection type, default type

CHAPTER THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX