CUBI-TK Documentation

Release 0+untagged.53.gd614fcc.dirty

Core Unit Bioinformatics

Jul 04, 2023

Installation Getting Started

Installation & Getting Started Instructions for the installation of the module and some examples to get you started.

Manual This section contains manuals for specific commands.

Use cases Use cases for common processing tasks.

Project Info More information on the project, including the changelog, list of contributing authors, and contribution instructions.

CHAPTER 1

Installation

Prerequisites when using conda:

```
$ conda create -n cubi-tk python=3.10
$ conda activate cubi-tk
```

Clone CUBI-TK and install:

```
$ git clone git@cubi-gitlab.bihealth.org:CUBI/Pipelines/cubi-tk.git
$ cd cubi-tk
$ pip install -e .
```

For building the manual or running tests you will need some more packages.

```
$ pip install -r requirements/develop.txt
```

1.1 Run tests

```
$ make test
```

1.2 Build manual

```
$ cd docs_manual
$ make clean html
```

CHAPTER 2

Command Line Interface

```
usage: cubi-tk [-h] [--verbose] [--version] [--config CONFIG]
               [--sodar-server-url SODAR_SERVER_URL]
               [--sodar-api-token SODAR_API_TOKEN]
               {isa-tpl,isa-tab,snappy,sodar,irods,org-raw,sea-snap,archive}
               ...
```

2.1 Positional Arguments

cmd Possible choices: isa-tpl, isa-tab, snappy, sodar, irods, org-raw, sea-snap, archive

2.2 Named Arguments

--verbose Increase verbosity.

Default: False

--version show program's version number and exit

2.3 Basic Configuration

--config Path to configuration file.

--sodar-server-url SODAR server URL key to use, defaults to env SODAR_SERVER_URL.

--sodar-api-token SODAR API token to use, defaults to env SODAR_API_TOKEN.

2.4 Sub-commands

2.4.1 isa-tpl

Create of ISA-tab directories from predefined templates.

```
cubi-tk isa-tpl [-h]
                {single_cell_rnaseq,bulk_rnaseq,tumor_normal_dna,tumor_normal_
↪triplets,germline,generic,microarray,ms_meta_biocrates,stem_cell_core_bulk,stem_
↪cell_core_sc}
                ...
```

Positional Arguments

tpl Possible choices: single_cell_rnaseq, bulk_rnaseq, tumor_normal_dna, tumor_normal_triplets, germline, generic, microarray, ms_meta_biocrates, stem_cell_core_bulk, stem_cell_core_sc

Sub-commands

single_cell_rnaseq

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl single_cell_rnaseq [-h]
                                   [--var-investigation-title VAR_INVESTIGATION_TITLE]
                                   [--var-sample-names VAR_SAMPLE_NAMES]
                                   [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                                   [--var-lib-kit VAR_LIB_KIT]
                                   [--var-source-meta VAR_SOURCE_META]
                                   [--var-sample-meta VAR_SAMPLE_META]
                                   [--var-dissociation-meta VAR_DISSOCIATION_META]
                                   [--var-library-construction-meta VAR_LIBRARY_
↪CONSTRUCTION_META]
                                   [--var-sequencing-meta VAR_SEQUENCING_META]
                                   [--var--library-types VAR__LIBRARY_TYPES]
                                   [--var-library-type VAR_LIBRARY_TYPE]
                                   [--var-sample-multiplexing VAR_SAMPLE_MULTIPLEXING]
                                   [--var-genotype-multiplexing VAR_GENOTYPE_
↪MULTIPLEXING]
                                   [--var-study-title VAR_STUDY_TITLE]
                                   [--var-s-file-name VAR_S_FILE_NAME]
                                   [--var-assay-prefix VAR_ASSAY_PREFIX]
                                   [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                                   [--var-assay-name VAR_ASSAY_NAME]
                                   [--var---output-dir VAR___OUTPUT_DIR]
                                  output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-lib-kit template variables 'lib_kit'

--var-source-meta template variables 'source_meta'

--var-sample-meta template variables 'sample_meta'

--var-dissociation-meta template variables 'dissociation_meta'

--var-library-construction-meta template variables 'library_construction_meta'

--var-sequencing-meta template variables 'sequencing_meta'

--var--library-types template variables '_library_types'

--var-library-type template variables 'library_type'

--var-sample-multiplexing template variables 'sample_multiplexing'

--var-genotype-multiplexing template variables 'genotype_multiplexing'

--var-study-title template variables 'study_title'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-technology-type template variables 'a_technology_type'

--var-assay-name template variables 'assay_name'

--var---output-dir template variables '__output_dir'

bulk_rnaseq

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl bulk_rnaseq [-h]
                            [--var-investigation-title VAR_INVESTIGATION_TITLE]
                            [--var-sample-names VAR_SAMPLE_NAMES]
                            [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                            [--var-a-measurement-types VAR_A_MEASUREMENT_TYPES]
                            [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                            [--var-a-technology-types VAR_A_TECHNOLOGY_TYPES]
                            [--var-lib-kit VAR_LIB_KIT]
                            [--var-organism VAR_ORGANISM]
                            [--var-batch VAR_BATCH]
                            [--var-lib-kits VAR_LIB_KITS]
                            [--var-organisms VAR_ORGANISMS]
                            [--var-instrument VAR_INSTRUMENT]
                            [--var-center-name VAR_CENTER_NAME]
                            [--var-center-contact VAR_CENTER_CONTACT]
                            [--var-study-title VAR_STUDY_TITLE]
                            [--var-s-file-name VAR_S_FILE_NAME]
                            [--var-assay-prefix VAR_ASSAY_PREFIX]
                            [--var-a-measurement-abbreviation VAR_A_MEASUREMENT_
↪ABBREVIATION]
```

(continues on next page)

(continued from previous page)

```
                                [--var-assay-name VAR_ASSAY_NAME]
                                [--var-sample-type VAR_SAMPLE_TYPE]
                                [--var-lib-strategy VAR_LIB_STRATEGY]
                                [--var-lib-selection VAR_LIB_SELECTION]
                                [--var-lib-layout VAR_LIB_LAYOUT]
                                [--var---output-dir VAR___OUTPUT_DIR]
                                output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-a-measurement-types template variables 'a_measurement_types'

--var-a-technology-type template variables 'a_technology_type'

--var-a-technology-types template variables 'a_technology_types'

--var-lib-kit template variables 'lib_kit'

--var-organism template variables 'organism'

--var-batch template variables 'batch'

--var-lib-kits template variables 'lib_kits'

--var-organisms template variables 'organisms'

--var-instrument template variables 'instrument'

--var-center-name template variables 'center_name'

--var-center-contact template variables 'center_contact'

--var-study-title template variables 'study_title'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-measurement-abbreviation template variables 'a_measurement_abbreviation'

--var-assay-name template variables 'assay_name'

--var-sample-type template variables 'sample_type'

--var-lib-strategy template variables 'lib_strategy'

--var-lib-selection template variables 'lib_selection'

--var-lib-layout template variables 'lib_layout'

--var---output-dir template variables '__output_dir'

tumor_normal_dna

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl tumor_normal_dna [-h]
                                 [--var-investigation-title VAR_INVESTIGATION_TITLE]
                                 [--var-sample-names VAR_SAMPLE_NAMES]
                                 [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                                 [--var-lib-kit VAR_LIB_KIT]
                                 [--var-lib-kits VAR_LIB_KITS]
                                 [--var-instrument VAR_INSTRUMENT]
                                 [--var-center-name VAR_CENTER_NAME]
                                 [--var-center-contact VAR_CENTER_CONTACT]
                                 [--var-study-title VAR_STUDY_TITLE]
                                 [--var-is-triplet VAR_IS_TRIPLET]
                                 [--var-s-file-name VAR_S_FILE_NAME]
                                 [--var-assay-prefix VAR_ASSAY_PREFIX]
                                 [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                                 [--var-a-measurement-abbreviation VAR_A_MEASUREMENT_
↪ABBREVIATION]
                                 [--var-assay-name VAR_ASSAY_NAME]
                                 [--var-sample-type VAR_SAMPLE_TYPE]
                                 [--var-lib-strategy VAR_LIB_STRATEGY]
                                 [--var-lib-selection VAR_LIB_SELECTION]
                                 [--var-lib-layout VAR_LIB_LAYOUT]
                                 [--var---output-dir VAR___OUTPUT_DIR]
                                 output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables ‘investigation_title’

--var-sample-names template variables ‘sample_names’

--var-a-measurement-type template variables ‘a_measurement_type’

--var-lib-kit template variables ‘lib_kit’

--var-lib-kits template variables ‘lib_kits’

--var-instrument template variables ‘instrument’

--var-center-name template variables ‘center_name’

--var-center-contact template variables ‘center_contact’

--var-study-title template variables ‘study_title’

--var-is-triplet template variables ‘is_triplet’

--var-s-file-name template variables ‘s_file_name’

--var-assay-prefix template variables ‘assay_prefix’

--var-a-technology-type template variables ‘a_technology_type’

--var-a-measurement-abbreviation template variables 'a_measurement_abbreviation'

--var-assay-name template variables 'assay_name'

--var-sample-type template variables 'sample_type'

--var-lib-strategy template variables 'lib_strategy'

--var-lib-selection template variables 'lib_selection'

--var-lib-layout template variables 'lib_layout'

--var---output-dir template variables '__output_dir'

tumor_normal_triplets

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl tumor_normal_triplets [-h]
                                      [--var-investigation-title VAR_INVESTIGATION_
↪TITLE]
                                      [--var-sample-names VAR_SAMPLE_NAMES]
                                      [--var-a-measurement-type VAR_A_MEASUREMENT_
↪TYPE]
                                      [--var-lib-kit VAR_LIB_KIT]
                                      [--var-lib-kits VAR_LIB_KITS]
                                      [--var-instrument VAR_INSTRUMENT]
                                      [--var-center-name VAR_CENTER_NAME]
                                      [--var-center-contact VAR_CENTER_CONTACT]
                                      [--var-study-title VAR_STUDY_TITLE]
                                      [--var-is-triplet VAR_IS_TRIPLET]
                                      [--var-s-file-name VAR_S_FILE_NAME]
                                      [--var-assay-prefix VAR_ASSAY_PREFIX]
                                      [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                                      [--var-a-measurement-abbreviation VAR_A_
↪MEASUREMENT_ABBREVIATION]
                                      [--var-assay-name VAR_ASSAY_NAME]
                                      [--var-sample-type VAR_SAMPLE_TYPE]
                                      [--var-lib-strategy VAR_LIB_STRATEGY]
                                      [--var-lib-selection VAR_LIB_SELECTION]
                                      [--var-lib-layout VAR_LIB_LAYOUT]
                                      [--var---output-dir VAR___OUTPUT_DIR]
                                     output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-lib-kit template variables 'lib_kit'

--var-lib-kits template variables 'lib_kits'

--var-instrument template variables 'instrument'

--var-center-name template variables 'center_name'

--var-center-contact template variables 'center_contact'

--var-study-title template variables 'study_title'

--var-is-triplet template variables 'is_triplet'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-technology-type template variables 'a_technology_type'

--var-a-measurement-abbreviation template variables 'a_measurement_abbreviation'

--var-assay-name template variables 'assay_name'

--var-sample-type template variables 'sample_type'

--var-lib-strategy template variables 'lib_strategy'

--var-lib-selection template variables 'lib_selection'

--var-lib-layout template variables 'lib_layout'

--var---output-dir template variables '__output_dir'

germline

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl germline [-h]
                         [--var-investigation-title VAR_INVESTIGATION_TITLE]
                         [--var-sample-names VAR_SAMPLE_NAMES]
                         [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                         [--var-lib-kit VAR_LIB_KIT] [--var-batch VAR_BATCH]
                         [--var-lib-kits VAR_LIB_KITS]
                         [--var-instrument VAR_INSTRUMENT]
                         [--var-center-name VAR_CENTER_NAME]
                         [--var-center-contact VAR_CENTER_CONTACT]
                         [--var-study-title VAR_STUDY_TITLE]
                         [--var-s-file-name VAR_S_FILE_NAME]
                         [--var-assay-prefix VAR_ASSAY_PREFIX]
                         [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                         [--var-a-measurement-abbreviation VAR_A_MEASUREMENT_
↪ABBREVIATION]
                         [--var-assay-name VAR_ASSAY_NAME]
                         [--var-sample-type VAR_SAMPLE_TYPE]
                         [--var-lib-strategy VAR_LIB_STRATEGY]
                         [--var-lib-selection VAR_LIB_SELECTION]
                         [--var-lib-layout VAR_LIB_LAYOUT]
                         [--var---output-dir VAR___OUTPUT_DIR]
                         output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-lib-kit template variables 'lib_kit'

--var-batch template variables 'batch'

--var-lib-kits template variables 'lib_kits'

--var-instrument template variables 'instrument'

--var-center-name template variables 'center_name'

--var-center-contact template variables 'center_contact'

--var-study-title template variables 'study_title'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-technology-type template variables 'a_technology_type'

--var-a-measurement-abbreviation template variables 'a_measurement_abbreviation'

--var-assay-name template variables 'assay_name'

--var-sample-type template variables 'sample_type'

--var-lib-strategy template variables 'lib_strategy'

--var-lib-selection template variables 'lib_selection'

--var-lib-layout template variables 'lib_layout'

--var---output-dir template variables '__output_dir'

generic

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl generic [-h]
                        [--var-investigation-title VAR_INVESTIGATION_TITLE]
                        [--var-sample-names VAR_SAMPLE_NAMES]
                        [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                        [--var-a-measurement-types VAR_A_MEASUREMENT_TYPES]
                        [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                        [--var-a-technology-types VAR_A_TECHNOLOGY_TYPES]
                        [--var-lib-kit VAR_LIB_KIT]
                        [--var-organism VAR_ORGANISM] [--var-batch VAR_BATCH]
                        [--var-lib-kits VAR_LIB_KITS]
                        [--var-organisms VAR_ORGANISMS]
```

(continues on next page)

(continued from previous page)

```
                        [--var-instrument VAR_INSTRUMENT]
                        [--var-center-name VAR_CENTER_NAME]
                        [--var-center-contact VAR_CENTER_CONTACT]
                        [--var-study-title VAR_STUDY_TITLE]
                        [--var-s-file-name VAR_S_FILE_NAME]
                        [--var-assay-prefix VAR_ASSAY_PREFIX]
                        [--var-a-measurement-abbreviation VAR_A_MEASUREMENT_
↪ABBREVIATION]
                        [--var-assay-name VAR_ASSAY_NAME]
                        [--var-sample-type VAR_SAMPLE_TYPE]
                        [--var-lib-strategy VAR_LIB_STRATEGY]
                        [--var-lib-selection VAR_LIB_SELECTION]
                        [--var-lib-layout VAR_LIB_LAYOUT]
                        [--var---output-dir VAR___OUTPUT_DIR]
                        output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-a-measurement-types template variables 'a_measurement_types'

--var-a-technology-type template variables 'a_technology_type'

--var-a-technology-types template variables 'a_technology_types'

--var-lib-kit template variables 'lib_kit'

--var-organism template variables 'organism'

--var-batch template variables 'batch'

--var-lib-kits template variables 'lib_kits'

--var-organisms template variables 'organisms'

--var-instrument template variables 'instrument'

--var-center-name template variables 'center_name'

--var-center-contact template variables 'center_contact'

--var-study-title template variables 'study_title'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-measurement-abbreviation template variables 'a_measurement_abbreviation'

--var-assay-name template variables 'assay_name'

--var-sample-type template variables 'sample_type'

--var-lib-strategy template variables 'lib_strategy'

--var-lib-selection template variables 'lib_selection'

--var-lib-layout template variables 'lib_layout'

--var---output-dir template variables '__output_dir'

microarray

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl microarray [-h]
                           [--var-investigation-title VAR_INVESTIGATION_TITLE]
                           [--var-sample-names VAR_SAMPLE_NAMES]
                           [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                           [--var-organism VAR_ORGANISM]
                           [--var-organisms VAR_ORGANISMS]
                           [--var-technology-platform VAR_TECHNOLOGY_PLATFORM]
                           [--var-array-design-ref VAR_ARRAY_DESIGN_REF]
                           [--var-study-title VAR_STUDY_TITLE]
                           [--var-s-file-name VAR_S_FILE_NAME]
                           [--var-assay-prefix VAR_ASSAY_PREFIX]
                           [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                           [--var-assay-name VAR_ASSAY_NAME]
                           [--var-terms VAR_TERMS]
                           [--var---output-dir VAR___OUTPUT_DIR]
                           output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-sample-names template variables 'sample_names'

--var-a-measurement-type template variables 'a_measurement_type'

--var-organism template variables 'organism'

--var-organisms template variables 'organisms'

--var-technology-platform template variables 'technology_platform'

--var-array-design-ref template variables 'array_design_ref'

--var-study-title template variables 'study_title'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-a-technology-type template variables 'a_technology_type'

--var-assay-name template variables 'assay_name'

--var-terms template variables 'terms'

--var---output-dir template variables '__output_dir'

ms_meta_biocrates

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl ms_meta_biocrates [-h]
                                  [--var-investigation-title VAR_INVESTIGATION_TITLE]
                                  [--var-study-title VAR_STUDY_TITLE]
                                  [--var-study-id VAR_STUDY_ID]
                                  [--var-study-file-name VAR_STUDY_FILE_NAME]
                                  [--var-sample-names VAR_SAMPLE_NAMES]
                                  [--var-organism VAR_ORGANISM]
                                  [--var-organisms VAR_ORGANISMS]
                                  [--var-assay-measurement-type VAR_ASSAY_MEASUREMENT_
↪TYPE]
                                  [--var-assay-technology-type VAR_ASSAY_TECHNOLOGY_
↪TYPE]
                                  [--var-assay-technology-types VAR_ASSAY_TECHNOLOGY_
↪TYPES]
                                  [--var-biocrates-kit VAR_BIOCRATES_KIT]
                                  [--var-assay-prefix VAR_ASSAY_PREFIX]
                                  [--var-assay-name VAR_ASSAY_NAME]
                                  [--var-assay-measurement-abbreviation-LC VAR_ASSAY_
↪MEASUREMENT_ABBREVIATION_LC]
                                  [--var-assay-measurement-abbreviation-FIA VAR_ASSAY_
↪MEASUREMENT_ABBREVIATION_FIA]
                                  [--var-biocrates-metidq-version VAR_BIOCRATES_
↪METIDQ_VERSION]
                                  [--var-metaquac-version VAR_METAQUAC_VERSION]
                                  [--var-instrument VAR_INSTRUMENT]
                                  [--var-instruments VAR_INSTRUMENTS]
                                  [--var-chromatography-instrument VAR_CHROMATOGRAPHY_
↪INSTRUMENT]
                                  [--var---output-dir VAR___OUTPUT_DIR]
                                  output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-investigation-title template variables 'investigation_title'

--var-study-title template variables 'study_title'

--var-study-id template variables 'study_id'

--var-study-file-name template variables 'study_file_name'

--var-sample-names template variables 'sample_names'

--var-organism template variables 'organism'

--var-organisms template variables 'organisms'

--var-assay-measurement-type template variables 'assay_measurement_type'

--var-assay-technology-type template variables 'assay_technology_type'

--var-assay-technology-types template variables 'assay_technology_types'

--var-biocrates-kit template variables 'biocrates_kit'

--var-assay-prefix template variables 'assay_prefix'

--var-assay-name template variables 'assay_name'

--var-assay-measurement-abbreviation-LC template variables 'assay_measurement_abbreviation_LC'

--var-assay-measurement-abbreviation-FIA template variables 'assay_measurement_abbreviation_FIA'

--var-biocrates-metidq-version template variables 'biocrates_metidq_version'

--var-metaquac-version template variables 'metaquac_version'

--var-instrument template variables 'instrument'

--var-instruments template variables 'instruments'

--var-chromatography-instrument template variables 'chromatography_instrument'

--var---output-dir template variables '__output_dir'

stem_cell_core_bulk

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl stem_cell_core_bulk [-h] [--var-group VAR_GROUP]
                                    [--var-study-title VAR_STUDY_TITLE]
                                    [--var-sample-numbers VAR_SAMPLE_NUMBERS]
                                    [--var-investigation-title VAR_INVESTIGATION_
↪TITLE]
                                    [--var-source-type VAR_SOURCE_TYPE]
                                    [--var-cellline VAR_CELLLINE]
                                    [--var-cellculture-meta VAR_CELLCULTURE_META]
                                    [--var-model-type VAR_MODEL_TYPE]
                                    [--var-sample-meta VAR_SAMPLE_META]
                                    [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                                    [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                                    [--var-library-kit VAR_LIBRARY_KIT]
                                    [--var-library-construction-meta VAR_LIBRARY_
↪CONSTRUCTION_META]
                                    [--var-sequencing-meta VAR_SEQUENCING_META]
                                    [--var-s-file-name VAR_S_FILE_NAME]
                                    [--var-assay-prefix VAR_ASSAY_PREFIX]
                                    [--var-assay-name VAR_ASSAY_NAME]
                                    [--var---output-dir VAR___OUTPUT_DIR]
                                    output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-group template variables 'group'

--var-study-title template variables 'study_title'

--var-sample-numbers template variables 'sample_numbers'

--var-investigation-title template variables 'investigation_title'

--var-source-type template variables 'source_type'

--var-cellline template variables 'cellline'

--var-cellculture-meta template variables 'cellculture_meta'

--var-model-type template variables 'model_type'

--var-sample-meta template variables 'sample_meta'

--var-a-measurement-type template variables 'a_measurement_type'

--var-a-technology-type template variables 'a_technology_type'

--var-library-kit template variables 'library_kit'

--var-library-construction-meta template variables 'library_construction_meta'

--var-sequencing-meta template variables 'sequencing_meta'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-assay-name template variables 'assay_name'

--var---output-dir template variables '__output_dir'

stem_cell_core_sc

When specifying the –var-* argument, you can use JSON syntax. Failing to parse JSON will keep the string value.

```
cubi-tk isa-tpl stem_cell_core_sc [-h] [--var-group VAR_GROUP]
                                  [--var-study-title VAR_STUDY_TITLE]
                                  [--var-sample-names VAR_SAMPLE_NAMES]
                                  [--var-investigation-title VAR_INVESTIGATION_TITLE]
                                  [--var-source-type VAR_SOURCE_TYPE]
                                  [--var-cellline VAR_CELLLINE]
                                  [--var-cellculture-meta VAR_CELLCULTURE_META]
                                  [--var-model-type VAR_MODEL_TYPE]
                                  [--var-sample-meta VAR_SAMPLE_META]
                                  [--var-a-measurement-type VAR_A_MEASUREMENT_TYPE]
                                  [--var-a-technology-type VAR_A_TECHNOLOGY_TYPE]
                                  [--var-library-kit VAR_LIBRARY_KIT]
                                  [--var-dissociation-meta VAR_DISSOCIATION_META]
                                  [--var-library-construction-meta VAR_LIBRARY_
↪CONSTRUCTION_META]
                                  [--var-sequencing-meta VAR_SEQUENCING_META]
                                  [--var--library-types VAR__LIBRARY_TYPES]
                                  [--var-library-type VAR_LIBRARY_TYPE]
                                  [--var-sample-multiplexing VAR_SAMPLE_MULTIPLEXING]
                                  [--var-genotype-multiplexing VAR_GENOTYPE_
↪MULTIPLEXING]
```

(continues on next page)

(continued from previous page)

```
                                    [--var-s-file-name VAR_S_FILE_NAME]
                                    [--var-assay-prefix VAR_ASSAY_PREFIX]
                                    [--var-assay-name VAR_ASSAY_NAME]
                                    [--var---output-dir VAR___OUTPUT_DIR]
                                    output_dir
```

Positional Arguments

output_dir Path to output directory

Named Arguments

--var-group template variables 'group'

--var-study-title template variables 'study_title'

--var-sample-names template variables 'sample_names'

--var-investigation-title template variables 'investigation_title'

--var-source-type template variables 'source_type'

--var-cellline template variables 'cellline'

--var-cellculture-meta template variables 'cellculture_meta'

--var-model-type template variables 'model_type'

--var-sample-meta template variables 'sample_meta'

--var-a-measurement-type template variables 'a_measurement_type'

--var-a-technology-type template variables 'a_technology_type'

--var-library-kit template variables 'library_kit'

--var-dissociation-meta template variables 'dissociation_meta'

--var-library-construction-meta template variables 'library_construction_meta'

--var-sequencing-meta template variables 'sequencing_meta'

--var--library-types template variables '_library_types'

--var-library-type template variables 'library_type'

--var-sample-multiplexing template variables 'sample_multiplexing'

--var-genotype-multiplexing template variables 'genotype_multiplexing'

--var-s-file-name template variables 's_file_name'

--var-assay-prefix template variables 'assay_prefix'

--var-assay-name template variables 'assay_name'

--var---output-dir template variables '__output_dir'

2.4.2 isa-tab

ISA-tab tools besides templating.

```
cubi-tk isa-tab [-h] {add-ped,resolve-hpo,annotate,validate} ...
```

Positional Arguments

isa_tab_cmd Possible choices: add-ped, resolve-hpo, annotate, validate

Sub-commands

add-ped

Add records from PED file to ISA-tab

```
cubi-tk isa-tab add-ped [-h] [--sample-name-normalization {snappy,none}]
                        [--yes] [--dry-run] [--no-show-diff]
                        [--show-diff-side-by-side] [--batch-no BATCH_NO]
                        [--library-type {WES,WGS,Panel_seq}]
                        [--library-layout {SINGLE,PAIRED}]
                        [--library-kit LIBRARY_KIT]
                        [--library-kit-catalogue-id LIBRARY_KIT_CATALOGUE_ID]
                        [--platform PLATFORM]
                        [--instrument-model INSTRUMENT_MODEL]
                        investigation.tsv pedigree.ped
```

Positional Arguments

investigation.tsv Path to ISA-tab investigation file.

pedigree.ped Path to PLINK PED file with records to add.

Named Arguments

--sample-name-normalization Possible choices: snappy, none

Normalize sample names, default: snappy, choices: snappy, none

Default: "snappy"

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--no-show-diff, -D Don't show change when creating/updating sample sheets.

Default: True

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

--batch-no Value to set as the batch number.

Default: “.”

--library-type Possible choices: WES, WGS, Panel_seq

The library type.

Default: “WES”

--library-layout Possible choices: SINGLE, PAIRED

The library layout.

Default: “PAIRED”

--library-kit The library kit used.

Default: “”

--library-kit-catalogue-id The library kit catalogue ID.

Default: “”

--platform The string to use for the platform

Default: “ILLUMINA”

--instrument-model The string to use for the instrument model

Default: “”

resolve-hpo

Resolve HPO term lists to ISA-tab fragments

```
cubi-tk isa-tab resolve-hpo [-h] [--hpo-obo-url HPO_OBO_URL] [term_file]
```

Positional Arguments

term_file Path to ISA-tab investigation file.

Default: <_io.TextIOWrapper name=’<stdin>’ mode=’r’ encoding=’UTF-8’>

Named Arguments

--hpo-obo-url Default URL to OBO file.

Default: “http://purl.obolibrary.org/obo/hp.obo”

annotate

Add annotation from CSV file to ISA-tab

```
cubi-tk isa-tab annotate [-h] [--yes] [--dry-run] [--no-show-diff]
                         [--show-diff-side-by-side] [--force-update]
                         [--target-study s_study.tsv]
                         [--target-assay a_assay.tsv]
                         investigation.tsv annotation.tsv
```

Positional Arguments

investigation.tsv Path to ISA-tab investigation file.

annotation.tsv Path to annotation (TSV) file with information to add.

Named Arguments

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--no-show-diff, -D Don't show change when creating/updating sample sheets.

Default: True

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

--force-update Overwrite non-empty ISA-tab entries.

Default: False

--target-study, -s File name study to annotate. If not provided, first study in investigation is used.

--target-assay, -a File name of assay to annotate. If not provided, first assay in investigation is used.

validate

Validate ISA-tab

```
cubi-tk isa-tab validate [-h] [--show-duplicate-warnings] investigation.tsv
```

Positional Arguments

investigation.tsv Path to ISA-tab investigation file.

Named Arguments

--show-duplicate-warnings Show duplicated warnings, i.e. with same message and same category (False by default)

Default: False

2.4.3 snappy

Tools for supporting the SNAPPY pipeline.

```
cubi-tk snappy [-h]
               {check-local,check-remote,itransfer-raw-data,itransfer-ngs-mapping,
↪itransfer-variant-calling,itransfer-step,pull-sheets,pull-all-data,pull-processed-
↪data,pull-raw-data,varfish-upload,kickoff}
               ...
```

Positional Arguments

snappy_cmd Possible choices: check-local, check-remote, itransfer-raw-data, itransfer-ngs-mapping, itransfer-variant-calling, itransfer-step, pull-sheets, pull-all-data, pull-processed-data, pull-raw-data, varfish-upload, kickoff

Sub-commands

check-local

Check consistency within local sample sheet and between local sheets and files

```
cubi-tk snappy check-local [-h] [--tsv-shortcut {germline,cancer}]
                           [--base-path BASE_PATH]
                           [project_uuids [project_uuids ...]]
```

Positional Arguments

project_uuids UUID(s) from project(s) to check. Use all if not given.

Named Arguments

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: "germline"

--base-path Base path of project (contains 'ngs_mapping/' etc.), spiders up from biomedsheet_tsv and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

check-remote

Check consistency within remote sample sheet and files

```
cubi-tk snappy check-remote [-h] [--sodar-url SODAR_URL]
                            [--sodar-api-token SODAR_API_TOKEN]
                            [--tsv-shortcut {cancer,generic,germline}]
                            [--base-path BASE_PATH] [--md5]
                            [--assay-uuid ASSAY_UUID]
                            project_uuid
```

Positional Arguments

project_uuid UUID from Project to check.

Named Arguments

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

--tsv-shortcut Possible choices: cancer, generic, germline

The shortcut TSV schema to use.

Default: "germline"

--base-path Base path of project (contains 'ngs_mapping/' etc.), spiders up from biomedsheet_tsv and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--md5 Flag to indicate if local and remote MD5 files should be compared.

Default: False

--assay-uuid UUID from Assay to check. Used to specify target while dealing with multi-assay projects.

itransfer-raw-data

Transfer FASTQs into iRODS landing zone

```
cubi-tk snappy itransfer-raw-data [-h] [--sodar-url SODAR_URL]
                                  [--sodar-api-token SODAR_API_TOKEN]
                                  [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                                  [--tsv-shortcut {germline,cancer}]
                                  [--first-batch FIRST_BATCH]
                                  [--last-batch LAST_BATCH]
                                  [--base-path BASE_PATH]
                                  [--remote-dir-date REMOTE_DIR_DATE]
                                  [--remote-dir-pattern REMOTE_DIR_PATTERN]
                                  [--yes] [--validate-and-move]
                                  [--assay ASSAY]
                                  destination
```

Positional Arguments

destination UUID from Landing Zone or Project - where files will be moved to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: "germline"

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--base-path Base path of project (contains 'ngs_mapping/' etc.), defaults to current path.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: "2023-07-04"

--remote-dir-pattern Pattern to use for constructing remote pattern

Default: "{library_name}/{step}/{date}"

--yes Assume all answers are yes, e.g., will create or use existing available landing zones without asking.

Default: False

--validate-and-move After files are transferred to SODAR, it will proceed with validation and move.

Default: False

--assay UUID of assay to download data for.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

itransfer-ngs-mapping

Transfer ngs_mapping results into iRODS landing zone

```
cubi-tk snappy itransfer-ngs-mapping [-h] [--sodar-url SODAR_URL]
                                     [--sodar-api-token SODAR_API_TOKEN]
                                     [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                                     [--tsv-shortcut {germline,cancer}]
                                     [--first-batch FIRST_BATCH]
                                     [--last-batch LAST_BATCH]
                                     [--base-path BASE_PATH]
                                     [--remote-dir-date REMOTE_DIR_DATE]
                                     [--remote-dir-pattern REMOTE_DIR_PATTERN]
                                     [--yes] [--validate-and-move]
                                     [--assay ASSAY] [--mapper MAPPER]
                                     destination
```

Positional Arguments

destination UUID from Landing Zone or Project - where files will be moved to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: “germline”

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--base-path Base path of project (contains ‘ngs_mapping/’ etc.), defaults to current path.

Default: “/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual”

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: “2023-07-04”

--remote-dir-pattern Pattern to use for constructing remote pattern

Default: “{library_name}/{step}/{date}”

--yes Assume all answers are yes, e.g., will create or use existing available landing zones without asking.

Default: False

--validate-and-move After files are transferred to SODAR, it will proceed with validation and move.

Default: False

--assay UUID of assay to download data for.

--mapper Name of the mapper to transfer for, defaults to bwa.

Default: “bwa”

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

itransfer-variant-calling

Transfer variant_calling results into iRODS landing zone

```
cubi-tk snappy itransfer-variant-calling [-h] [--sodar-url SODAR_URL]
                                         [--sodar-api-token SODAR_API_TOKEN]
                                         [--num-parallel-transfers NUM_PARALLEL_
↪TRANSFERS]
                                         [--tsv-shortcut {germline,cancer}]
                                         [--first-batch FIRST_BATCH]
                                         [--last-batch LAST_BATCH]
                                         [--base-path BASE_PATH]
                                         [--remote-dir-date REMOTE_DIR_DATE]
                                         [--remote-dir-pattern REMOTE_DIR_PATTERN]
                                         [--yes] [--validate-and-move]
                                         [--assay ASSAY] [--mapper MAPPER]
                                         [--caller CALLER]
                                         destination
```

Positional Arguments

destination UUID from Landing Zone or Project - where files will be moved to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: "germline"

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--base-path Base path of project (contains 'ngs_mapping/' etc.), defaults to current path.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: "2023-07-04"

--remote-dir-pattern Pattern to use for constructing remote pattern

Default: "{library_name}/{step}/{date}"

--yes Assume all answers are yes, e.g., will create or use existing available landing zones without asking.

Default: False

--validate-and-move After files are transferred to SODAR, it will proceed with validation and move.

Default: False

--assay UUID of assay to download data for.

--mapper Name of the mapper to transfer for, defaults to bwa.

Default: "bwa"

--caller Name of the variant caller to transfer for, defaults to gatk_hc

Default: "gatk_hc"

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

itransfer-step

Transfer snappy step results into iRODS landing zone

```
cubi-tk snappy itransfer-step [-h] [--sodar-url SODAR_URL]
                              [--sodar-api-token SODAR_API_TOKEN]
                              [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                              [--tsv-shortcut {germline,cancer}]
                              [--first-batch FIRST_BATCH]
                              [--last-batch LAST_BATCH]
                              [--base-path BASE_PATH]
                              [--remote-dir-date REMOTE_DIR_DATE]
                              [--remote-dir-pattern REMOTE_DIR_PATTERN]
                              [--yes] [--validate-and-move] [--assay ASSAY]
                              [--step STEP] [--tool [TOOL [TOOL ...]]]
                              destination
```

Positional Arguments

destination UUID from Landing Zone or Project - where files will be moved to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: "germline"

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--base-path Base path of project (contains 'ngs_mapping/' etc.), defaults to current path.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: "2023-07-04"

--remote-dir-pattern Pattern to use for constructing remote pattern

Default: "{library_name}/{step}/{date}"

--yes Assume all answers are yes, e.g., will create or use existing available landing zones without asking.

Default: False

--validate-and-move After files are transferred to SODAR, it will proceed with validation and move.

Default: False

--assay UUID of assay to download data for.

--step Name of the snappy pipeline step (step name must be identical to step directory).Steps names are available from the snappy command snappy-start-step –help

--tool Name of the tool, for example bwa. Tools order in important:it must match the order used to generate filename prefix.For example, the variant annotation step requires the mapper, caller andthe annotator software. In that case, the snappy file prefix is:<mapper>.<caller>.<annotator>, so the command would be:–tool <mapper> <vcaller> <annotator>. Some steps add more information to theirprefix, for example 'jannovar_somatic_vcf'

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

pull-sheets

Pull SODAR sample sheets into biomedsheet

```
cubi-tk snappy pull-sheets [-h] [--base-path BASE_PATH] [--yes] [--dry-run]
                           [--no-show-diff] [--show-diff-side-by-side]
                           [--library-types LIBRARY_TYPES]
                           [--first-batch FIRST_BATCH]
                           [--last-batch LAST_BATCH]
                           [--tsv-shortcut {cancer,generic,germline}]
```

Named Arguments

--base-path Base path of project (contains '.snappy_pipeline/' etc.), spiders up from current work directory and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--no-show-diff, -D Don't show change when creating/updating sample sheets.

Default: True

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

--library-types Library type(s) to use, comma-separated, default is to use all.

--first-batch First batch to be included in local sample sheet. Defaults: 0.

Default: 0

--last-batch Last batch to be included in local sample sheet. Not used by default.

--tsv-shortcut Possible choices: cancer, generic, germline

The shortcut TSV schema to use; default: 'germline'.

Default: "germline"

pull-all-data

Pull all data from SODAR to specified output directory

```
cubi-tk snappy pull-all-data [-h] [--base-path BASE_PATH]
                             [--sodar-url SODAR_URL]
                             [--sodar-api-token SODAR_API_TOKEN]
                             --output-directory OUTPUT_DIRECTORY [--overwrite]
                             [--first-batch FIRST_BATCH] [--samples SAMPLES]
                             [--allow-missing] [--yes] [--dry-run]
```

(continues on next page)

(continued from previous page)

```
                                [--irsync-threads IRSYNC_THREADS]
                                [--assay ASSAY_UUID]
                                project_uuid
```

Positional Arguments

project_uuid UUID of project to download data for.

Named Arguments

--base-path Base path of project (contains '.snappy_pipeline/' etc.), spiders up from current work directory and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--output-directory Output directory, where downloaded files will be stored.

--overwrite Allow overwriting of files

Default: False

--first-batch First batch number to pull

Default: 0

--samples Optional list of samples to pull

--allow-missing Allow missing data in assay

Default: False

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--irsync-threads Parameter -N to pass to irsync

--assay UUID of assay to create landing zone for.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

pull-processed-data

Pull processed data from SODAR to specified output directory

```
cubi-tk snappy pull-processed-data [-h] [--sodar-url SODAR_URL]
                                   [--sodar-api-token SODAR_API_TOKEN]
                                   [--tsv-shortcut {cancer,generic,germline}]
                                   [--base-path BASE_PATH]
                                   [--selected-samples SELECTED_SAMPLES]
                                   [--first-batch FIRST_BATCH]
                                   [--last-batch LAST_BATCH]
                                   --output-directory OUTPUT_DIRECTORY
                                   [--sample-id] --file-type
                                   {bam,vcf,txt,csv,log}
                                   [--download-all-versions] [--overwrite]
                                   [--assay-uuid ASSAY_UUID]
                                   project_uuid
```

Positional Arguments

project_uuid UUID from Project to check.

Named Arguments

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

--tsv-shortcut Possible choices: cancer, generic, germline

The shortcut TSV schema to use; default: 'germline'.

Default: "germline"

--base-path Base path of project (contains 'ngs_mapping/' etc.), spiders up from biomedsheet_tsv and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--selected-samples Limits the request to the listed sample names. Don't include the full library name, just the sample name (e.g., 'P001' instead of 'P001-N1-DNA1-WES1'). Separate the sample with comma for multiple samples, example: 'P001,P002,P003'. Note: argument overrides batch related arguments.

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--output-directory Output directory, where downloaded files will be stored.

--sample-id Flag to indicate if search should be based on sample identifier (e.g.'P001') instead of library name (e.g. 'P001-N1-DNA1-WGS1').

Default: False

--file-type Possible choices: bam, vcf, txt, csv, log

File extensions to be retrieved. Valid options: ('bam', 'vcf', 'txt', 'csv', 'log')

--download-all-versions By default only the latest version of a file will be download. For instance, if a was uploaded two times, in '2022-01-31' and '2022-02-28', only the latest is downloaded. If this flag is present, both versions will be downloaded.

Default: False

--overwrite Allow overwriting of local files.

Default: False

--assay-uuid UUID from Assay to check. Used to specify target while dealing with multi-assay projects.

pull-raw-data

Pull raw data from SODAR to SNAPPY dataset raw data directory

```
cubi-tk snappy pull-raw-data [-h] [--base-path BASE_PATH]
                             [--sodar-url SODAR_URL]
                             [--sodar-api-token SODAR_API_TOKEN] [--dry-run]
                             [--overwrite]
                             [--tsv-shortcut {cancer,generic,germline}]
                             [--first-batch FIRST_BATCH]
                             [--last-batch LAST_BATCH] [--samples SAMPLES]
                             [--use-library-name] [--assay-uuid ASSAY_UUID]
                            project_uuid
```

Positional Arguments

project_uuid UUID of project to download data for.

Named Arguments

--base-path Base path of project (contains '.snappy_pipeline/' etc.), spiders up from current work directory and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--dry-run, -n Perform a dry run, i.e., just displays the files that would be downloaded.

Default: False

--overwrite Allow overwriting of files

Default: False

--tsv-shortcut Possible choices: cancer, generic, germline

The shortcut TSV schema to use.

Default: "germline"

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--samples Optional list of samples to pull

--use-library-name Flag to indicate that the search in SODAR directories should be based on library name (e.g. 'P001-N1-DNA1-WGS1') instead of sample identifier (e.g.'P001') in the file name.

Default: False

--assay-uuid UUID of assay to create landing zone for.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

varfish-upload

Upload variant analysis results into VarFish

```
cubi-tk snappy varfish-upload [-h] [--varfish-config VARFISH_CONFIG]
                              [--varfish-server-url VARFISH_SERVER_URL]
                              [--varfish-api-token VARFISH_API_TOKEN]
                              [--base-path BASE_PATH] [--steps STEPS]
                              [--external-data] [--min-batch MIN_BATCH]
                              [--yes] [--samples SAMPLES]
                              project [project ...]
```

Positional Arguments

project The UUID(s) of the SODAR project to submit.

Named Arguments

--base-path Base path of project (contains '.snappy_pipeline/' etc.), spiders up from current work directory and falls back to current working directory by default.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--steps Pipeline steps to consider for the export. Defaults to include all of the following; specify this with +name/-name to add/remove and either give multiple arguments or use a comma-separated list. {ngs_mapping, targeted_seq_cnv_export, varfish_export, variant_export, variant_export_external, wgs_cnv_export, wgs_cnv_export_external, wgs_sv_export, wgs_sv_export_external}

Default: []

--external-data Flag to indicate that data was externally generated. Search for files will not filter based on common internally tool combinations, example: 'bwa.delly2' or 'bwa.gatk_hc'.

Default: False

--min-batch Smallest batch to transfer, keep empty to transfer all.

--yes, -y Assume yes to all answers

Default: False

--samples Limits the submission to the listed sample names. Don't include the full library name just the sample name (e.g., 'P001' instead of 'P001-N1-DNA1-WES1'). Separate the sample with comma for multiple samples, example: 'P001,P002,P003'.

Default: ""

VarFish Configuration

--varfish-config Path to configuration file.

--varfish-server-url SODAR server URL key to use, defaults to env VARFISH_SERVER_URL.

--varfish-api-token SODAR API token to use, defaults to env VARFISH_API_TOKEN.

kickoff

Kick-off SNAPPY pipeline steps.

```
cubi-tk snappy kickoff [-h] [--dry-run] [--timeout TIMEOUT] [path]
```

Positional Arguments

path Path into SNAPPY directory (below a directory containing .snappy_pipeline).

Named Arguments

--dry-run, -n Perform dry-run, do not do anything.

Default: False

--timeout Number of seconds to wait for commands.

Default: 10

2.4.4 sodar

SODAR command line interface.

```
cubi-tk sodar [-h]
              {add-ped,download-sheet,upload-sheet,pull-raw-data,landing-zone-create,
↪landing-zone-list,landing-zone-move,ingest-fastq,check-remote}
              ...
```

Positional Arguments

sodar_cmd Possible choices: add-ped, download-sheet, upload-sheet, pull-raw-data, landing-zone-create, landing-zone-list, landing-zone-move, ingest-fastq, check-remote

Sub-commands

add-ped

Augment sample sheet from PED file

```
cubi-tk sodar add-ped [-h] [--sodar-url SODAR_URL]
                      [--sodar-api-token SODAR_API_TOKEN] [--dry-run]
                      [--show-diff] [--show-diff-side-by-side]
                      [--sample-name-normalization {snappy,none}] [--yes]
                      [--batch-no BATCH_NO]
                      [--library-type {WES,WGS,Panel_seq}]
                      [--library-layout {SINGLE,PAIRED}]
                      [--library-kit LIBRARY_KIT]
                      [--library-kit-catalogue-id LIBRARY_KIT_CATALOGUE_ID]
                      [--platform PLATFORM]
                      [--instrument-model INSTRUMENT_MODEL]
                      project_uuid pedigree.ped
```

Positional Arguments

project_uuid UUID of project to download the ISA-tab for.

pedigree.ped Path to PLINK PED file with records to add.

Named Arguments

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--show-diff, -D Show change when creating/updating sample sheets.

Default: False

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

--sample-name-normalization Possible choices: snappy, none

Normalize sample names, default: snappy, choices: snappy, none

Default: "snappy"

--yes Assume all answers are yes.

Default: False

--batch-no Value to set as the batch number.

Default: "."

--library-type Possible choices: WES, WGS, Panel_seq

The library type.

Default: "WES"

--library-layout Possible choices: SINGLE, PAIRED

The library layout.

Default: "PAIRED"

--library-kit The library kit used.

Default: ""

--library-kit-catalogue-id The library kit catalogue ID.

Default: ""

--platform The string to use for the platform

Default: "ILLUMINA"

--instrument-model The string to use for the instrument model

Default: ""

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

download-sheet

Download ISA-tab

```
cubi-tk sodar download-sheet [-h] [--sodar-url SODAR_URL]
                             [--sodar-api-token SODAR_API_TOKEN]
                             [--no-makedirs] [--overwrite] [--yes] [--dry-run]
                             [--show-diff] [--show-diff-side-by-side]
                             project_uuid output_dir
```

Positional Arguments

project_uuid UUID of project to download the ISA-tab for.

output_dir Path to output directory to write the sheet to.

Named Arguments

--no-makedirs Create output directories

Default: True

--overwrite Allow overwriting of files

Default: False

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--show-diff, -D Show change when creating/updating sample sheets.

Default: False

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

upload-sheet

Upload and replace ISA-tab

```
cubi-tk sodar upload-sheet [-h] [--sodar-url SODAR_URL]
                           [--sodar-api-token SODAR_API_TOKEN]
                           project_uuid input_investigation_file
```

Positional Arguments

project_uuid UUID of project to upload the ISA-tab for.

input_investigation_file Path to input investigation file.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

pull-raw-data

Download raw data from iRODS

```
cubi-tk sodar pull-raw-data [-h] [--sodar-url SODAR_URL]
                            [--sodar-api-token SODAR_API_TOKEN] [--overwrite]
                            [--min-batch MIN_BATCH] [--allow-missing] [--yes]
                            [--dry-run] [--irsync-threads IRSYNC_THREADS]
                            [--assay ASSAY]
                            project_uuid output_dir
```

Positional Arguments

project_uuid UUID of project to download data for.

output_dir Path to output directory to write the raw data to.

Named Arguments

--overwrite Allow overwriting of files

Default: False

--min-batch Minimal batch number to pull

Default: 0

--allow-missing Allow missing data in assay

Default: False

--yes Assume all answers are yes.

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--irsync-threads Parameter -N to pass to irsync

--assay UUID of assay to download data for.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

landing-zone-create

Creating landing zone

```
cubi-tk sodar landing-zone-create [-h] [--sodar-url SODAR_URL]
                                  [--sodar-api-token SODAR_API_TOKEN]
                                  [--unless-exists] [--dry-run]
                                  [--assay ASSAY] [--format FORMAT_STRING]
                                  project_uuid
```

Positional Arguments

project_uuid UUID of project to create the landing zone in.

Named Arguments

--unless-exists If there already is a landing zone in the current project then use this one

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--assay UUID of assay to create landing zone for.

--format Format string for printing, e.g. %(uuid)s

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

landing-zone-list

List landing zones

```
cubi-tk sodar landing-zone-list [-h] [--sodar-url SODAR_URL]
                                [--sodar-api-token SODAR_API_TOKEN]
                                [--unless-exists] [--dry-run]
                                [--format FORMAT_STRING]
                                project_uuid
```

Positional Arguments

project_uuid UUID of project to create the landing zone in.

Named Arguments

--unless-exists If there already is a landing zone in the current project then use this one

Default: False

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--format Format string for printing, e.g. %(uuid)s

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

landing-zone-move

Submit landing zone for moving

```
cubi-tk sodar landing-zone-move [-h] [--sodar-url SODAR_URL]
                                [--sodar-api-token SODAR_API_TOKEN]
                                [--dry-run] [--format FORMAT_STRING]
                                landing_zone_uuid
```

Positional Arguments

landing_zone_uuid UUID of landing zone to move.

Named Arguments

--dry-run, -n Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--format Format string for printing, e.g. %(uuid)s

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

ingest-fastq

Upload external files to SODAR (defaults for fastq)

```
cubi-tk sodar ingest-fastq [-h] [--sodar-url SODAR_URL]
                           [--sodar-api-token SODAR_API_TOKEN]
                           [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                           [--yes] [--base-path BASE_PATH]
                           [--remote-dir-date REMOTE_DIR_DATE]
                           [--src-regex SRC_REGEX]
                           [--remote-dir-pattern REMOTE_DIR_PATTERN]
                           [--add-suffix ADD_SUFFIX] [-m MATCH REPL]
                           [--tmp TMP]
                           sources [sources ...] destination
```

Positional Arguments

sources paths to fastq folders

destination UUID or iRods path of landing zone to move to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--yes Assume the answer to all prompts is 'yes'

Default: False

--base-path Base path of project (contains 'ngs_mapping/' etc.), defaults to current path.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: "2023-07-04"

--src-regex Regular expression to use for matching input fastq files, default: (.*/)?(?P<sample>.+?)(?:_(?P<lane>L[0-9]+?))?(?:_(?P<mate>R[0-9]+?))?(?:_(?P<batch>[0-9]+?))?.f(?:ast)?q.gz

Default: "(.*/)?(?P<sample>.+?)(?:_(?P<lane>L[0-9]+?))?(?:_(?P<mate>R[0-9]+?))?(?:_(?P<batch>[0-9]+?))?.f(?:ast)?q.gz"

--remote-dir-pattern Pattern to use for constructing remote pattern, default: {sample}/{date}/{filename}

Default: "{sample}/{date}/{filename}"

--add-suffix Suffix to add to all file names (e.g. '-N1-DNA1-WES1').

Default: ""

-m, --remote-dir-mapping Substitutions applied to the filled remote dir paths. Can for example be used to modify sample names. Use pythons regex syntax of 're.sub' package. This argument can be used multiple times (i.e. '-m <regex1> <repl1> -m <regex2> <repl2>' ...).

Default: []

--tmp Folder to save files from WebDAV temporarily, if set as source.

Default: "temp/"

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

check-remote

Compare local files with md5 sum against SODAR/iRODS

```
cubi-tk sodar check-remote [-h] [--sodar-url SODAR_URL]
                           [--sodar-api-token SODAR_API_TOKEN] [-p BASE_PATH]
                           [--filename-only] [--recheck-md5]
                           [--report-md5sums] [--assay-uuid ASSAY_UUID]
                           project_uuid
```

Positional Arguments

project_uuid UUID from Project to check.

Named Arguments

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

-p, --base-path Base path in which local files with md5 sums should be identified. Default: CWD

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--filename-only Flag to indicate whether file comparison between local and remote files should only use file names and ignore md5 values.

Default: False

--recheck-md5 Flag to double check that md5 sums stored in local files do actually match their corresponding files

Default: False

--report-md5sums Flag to indicate if md5 sums should be included in file report

Default: False

--assay-uuid UUID from Assay to check. Used to specify target while dealing with multi-assay projects.

2.4.5 irods

iRods command line interface.

```
cubi-tk irods [-h] {check} ...
```

Positional Arguments

irods_cmd Possible choices: check

Sub-commands

check

Check target iRODS collection (all MD5 files? metadata MD5 consistent? enough replicas?).

```
cubi-tk irods check [-h] [-r REQ_NUM_REPS] [-p NUM_PARALLEL_TESTS]
                    [-d NUM_DISPLAY_FILES] [-s HASH_SCHEME]
                    irods_path
```

Positional Arguments

irods_path Path to an iRODS collection.

Named Arguments

-r, --num-replicas Minimum number of replicas, defaults to 2

Default: 2

-p, --num-parallel-tests Number of parallel tests, defaults to 4

Default: 4

-d, --num-display-files Number of files listed when checking, defaults to 20

Default: 20

-s, --hash-scheme Hash scheme used to verify checksums, defaults to MD5

Default: "MD5"

2.4.6 org-raw

org_raw command line interface.

```
cubi-tk org-raw [-h] {check,organize} ...
```

Positional Arguments

org_raw_cmd Possible choices: check, organize

Sub-commands

check

Check consistency of raw data

```
cubi-tk org-raw check [-h] [--num-threads NUM_THREADS] [--no-gz-check]
                      [--no-md5-check] [--no-compute-md5]
                      [--missing-md5-error] [--create-md5-fail-no-error]
                      FILE.fastq.gz [FILE.fastq.gz ...]
```

Positional Arguments

FILE.fastq.gz Path(s) to .fastq.gz files to perform the check for

Named Arguments

--num-threads Number of parallel threads

Default: 0

--no-gz-check Deactivate check for gzip consistency (default is to perform check).

Default: True

--no-md5-check Deactivate comparison of MD5 sum if .md5 file exists (default is to perform check).

Default: True

--no-compute-md5 Deactivate computation of MD5 sum if missing (default is to compute MD5 sum).

Default: True

--missing-md5-error Make missing .md5 files constitute an error. Default is to issue an log message only.

Default: False

--create-md5-fail-no-error Make failure to create .md5 file not an error. Default is to make it an error.

Default: True

organize

Check consistency of raw data

```
cubi-tk org-raw organize [-h] [--dry-run] [--yes] [--move] [--no-check]
                         [--src-regex SRC_REGEX] [--dest-pattern DEST_PATTERN]
                         [--num-threads NUM_THREADS] [--no-gz-check]
                         [--no-md5-check] [--no-compute-md5]
                         [--missing-md5-error] [--create-md5-fail-no-error]
                         out_path path.fastq.gz [path.fastq.gz ...]
```

Positional Arguments

out_path Path to output directory.

path.fastq.gz Path to input files.

Named Arguments

--dry-run Dry-run, do not actually do anything

Default: False

--yes Assume the answer to all prompts is 'yes'

Default: False

--move Move file(s) instead of copying, default is to copy.

Default: False

--no-check Do not run 'raw-org check' on output (default is to run).

Default: True

--src-regex Regular expression for parsing file paths. Default: (.*/)?(?P<sample>.+)(?:-.+?)?.f(?:ast)?q.gz

Default: "(.*/)?(?P<sample>.+)(?:-.+?)?.f(?:ast)?q.gz"

--dest-pattern Format expression for destination path generation. Default: {sample_name}/{file_name}

Default: "{sample_name}/{file_name}"

--num-threads Number of parallel threads

Default: 0

--no-gz-check Deactivate check for gzip consistency (default is to perform check).

Default: True

--no-md5-check Deactivate comparison of MD5 sum if .md5 file exists (default is to perform check).

Default: True

--no-compute-md5 Deactivate computation of MD5 sum if missing (default is to compute MD5 sum).

Default: True

--missing-md5-error Make missing .md5 files constitute an error. Default is to issue an log message only.

Default: False

--create-md5-fail-no-error Make failure to create .md5 file not an error. Default is to make it an error.

Default: True

2.4.7 sea-snap

Tools for supporting the RNA-SeASnaP pipeline.

```
cubi-tk sea-snap [-h]
                 {itransfer-raw-data,itransfer-results,working-dir,write-sample-info,
↪check-irods}
                 ...
```

Positional Arguments

sea_snap_cmd Possible choices: itransfer-raw-data, itransfer-results, working-dir, write-sample-info, check-irods

Sub-commands

itransfer-raw-data

Transfer FASTQs into iRODS landing zone

```
cubi-tk sea-snap itransfer-raw-data [-h] [--sodar-url SODAR_URL]
                                    [--sodar-api-token SODAR_API_TOKEN]
                                    [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                                    [--tsv-shortcut {germline,cancer}]
                                    [--first-batch FIRST_BATCH]
                                    [--last-batch LAST_BATCH]
                                    [--base-path BASE_PATH]
```

(continues on next page)

(continued from previous page)

```
                                    [--remote-dir-date REMOTE_DIR_DATE]
                                    [--remote-dir-pattern REMOTE_DIR_PATTERN]
                                    [--yes] [--validate-and-move]
                                    [--assay ASSAY]
                                    destination
```

Positional Arguments

destination UUID from Landing Zone or Project - where files will be moved to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

--tsv-shortcut Possible choices: germline, cancer

The shortcut TSV schema to use.

Default: "germline"

--first-batch First batch to be transferred. Defaults: 0.

Default: 0

--last-batch Last batch to be transferred.

--base-path Base path of project (contains 'ngs_mapping/' etc.), defaults to current path.

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

--remote-dir-date Date to use in remote directory, defaults to YYYY-MM-DD of today.

Default: "2023-07-04"

--remote-dir-pattern Pattern to use for constructing remote pattern

Default: "{library_name}/{step}/{date}"

--yes Assume all answers are yes, e.g., will create or use existing available landing zones without asking.

Default: False

--validate-and-move After files are transferred to SODAR, it will proceed with validation and move.

Default: False

--assay UUID of assay to download data for.

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

itransfer-results

Transfer mapping results into iRODS landing zone

```
cubi-tk sea-snap itransfer-results [-h] [--sodar-url SODAR_URL]
                                   [--sodar-api-token SODAR_API_TOKEN]
                                   [--num-parallel-transfers NUM_PARALLEL_TRANSFERS]
                                   transfer_blueprint destination
```

Positional Arguments

transfer_blueprint Path to blueprint file to load. This file contains commands to sync files with iRODS. Blocks of commands separated by an empty line will be executed together in one thread.

destination UUID or iRods path of landing zone to move to.

Named Arguments

--num-parallel-transfers Number of parallel transfers, defaults to 8

Default: 8

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-api-token Authentication token when talking to SODAR. Defaults to SODAR_API_TOKEN environment variable.

working-dir

Create working directory

```
cubi-tk sea-snap working-dir [-h] [--dry-run] [--dirname DIRNAME]
                             [--configs {mapping,DE} [{mapping,DE} ...]]
                             [sea_snap_path]
```

Positional Arguments

sea_snap_path Path into RNA-SeA-SnaP directory (below a directory containing 'mapping_pipeline.snake').

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/docs_manual"

Named Arguments

--dry-run, -n Perform dry-run, do not do anything.

Default: False

--dirname, -d Name of the working directory to create (default: 'results_YEAR_MONTH_DAY/').

Default: "results_%Y_%m_%d/"

--configs, -c Possible choices: mapping, DE

Configs to be imported (default: all).

Default: ['mapping', 'DE']

write-sample-info

Generate sample info file

```
cubi-tk sea-snap write-sample-info [-h] [--allow-overwrite] [--dry-run]
                                   [--show-diff] [--show-diff-side-by-side]
                                   [--from-file FROM_FILE]
                                   [--isa-assay ISA_ASSAY]
                                   [--project_uuid PROJECT_UUID]
                                   [--output_folder OUTPUT_FOLDER]
                                   [--overwrite-isa] [--sodar-url SODAR_URL]
                                   [--sodar-auth-token SODAR_AUTH_TOKEN]
                                  in_path_pattern [output_file]
```

Positional Arguments

in_path_pattern Path pattern to use for extracting input file information. See https://cubi-gitlab.bihealth.org/CUBI/Pipelines/sea-snap/blob/master/documentation/prepare_input.md#fastq-files-folder-structure.

output_file Filename ending with '.yaml' or '.tsv'. default: sample_info.yaml.

Default: sample_info.yaml

Named Arguments

--allow-overwrite Allow to overwrite output file, default is not to allow overwriting output file.

Default: False

--dry-run Perform a dry run, i.e., don't change anything only display change, implies '–show-diff'.

Default: False

--show-diff Show change when creating/updating sample sheets.

Default: False

--show-diff-side-by-side Show diff side by side instead of unified.

Default: False

--from-file Path to yaml file to convert to tsv or tsv to yaml. Not used, if not specified.

--isa-assay Path to ISA assay file. Not used, if not specified.

pull ISA files

--project_uuid If set pull ISA files from SODAR. UUID of project to pull from.

Default: False

--output_folder Output folder path to store ISA files.

Default: "ISA_files/"

--overwrite-isa Allow to overwrite output file, default is not to allow overwriting output file.

Default: False

SODAR-related

--sodar-url URL to SODAR, defaults to SODAR_URL environment variable or fallback to https://sodar.bihealth.org/

Default: "https://sodar.bihealth.org/"

--sodar-auth-token Authentication token when talking to SODAR. Defaults to SODAR_AUTH_TOKEN environment variable.

check-irods

Check consistency of sample info, blueprint and files on SODAR

```
cubi-tk sea-snap check-irods [-h] [--num-replicas NUM_REPLICAS]
                             [--num-parallel-tests NUM_PARALLEL_TESTS] [--yes]
                             [--transfer-blueprint TRANSFER_BLUEPRINT]
                             results_folder irods_path
```

Positional Arguments

results_folder Path to a Sea-snap results folder.

irods_path Path to an iRods collection.

Named Arguments

--num-replicas Minimum number of replicas, defaults to 2

Default: 2

--num-parallel-tests Number of parallel tests, defaults to 8

Default: 8

--yes Assume the answer to all prompts is 'yes'

Default: False

--transfer-blueprint Filename of blueprint file for export to SODAR (created e.g. with './seasnap sc l export'). Assumed to be in the results folder. Default: 'SODAR_export_blueprint.txt'

Default: "SODAR_export_blueprint.txt"

2.4.8 archive

helper for archiving projects.

```
cubi-tk archive [-h] {copy,prepare,readme,summary} ...
```

Positional Arguments

archive_cmd Possible choices: copy, prepare, readme, summary

Sub-commands

copy

Perform archival (copy and audit)

```
cubi-tk archive copy [-h] [--num-threads NUM_THREADS]
                     [--skip [SKIP [SKIP ...]]] [--keep-workdir-hashdeep]
                     [--read-only]
                     project destination
```

Positional Arguments

project Path of project directory

destination Final destination directory for archive, must not exist

Named Arguments

--num-threads Number of parallel threads

Default: 4

--skip Step to skip (hashdeep, rsync, audit)

--keep-workdir-hashdeep Save hashdeep report & audit of the temporary destination

Default: False

--read-only Change destination files to read-only

Default: False

prepare

Prepare the project directory for archival

```
cubi-tk archive prepare [-h] [--num-threads NUM_THREADS] [--rules RULES]
                        [--readme README] [--ignore-tar-errors]
                        project destination
```

Positional Arguments

project	Path of project directory
destination	Destination directory (for symlinks and later archival)

Named Arguments

--num-threads	Number of parallel threads Default: 4
--rules, -r	Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/cubi_tk/archive/default_rules.yaml"
--readme	Path to README.md created with cubi-tk
--ignore-tar-errors	Ignore errors due to access permissions in when compressind folders Default: False

readme

Prepare a valid README.md

```
cubi-tk archive readme [-h] [--skip-collect] [--is-valid]
                       [--var-directory VAR_DIRECTORY]
                       [--var-PI-name VAR_PI_NAME]
                       [--var-PI-email VAR_PI_EMAIL]
                       [--var-archiver-name VAR_ARCHIVER_NAME]
                       [--var-archiver-email VAR_ARCHIVER_EMAIL]
                       [--var-CUBI-name VAR_CUBI_NAME]
                       [--var-client-name VAR_CLIENT_NAME]
                       [--var-client-email VAR_CLIENT_EMAIL]
                       [--var-SODAR-UUID VAR_SODAR_UUID]
                       [--var-SODAR-URL VAR_SODAR_URL]
                       [--var-Gitlab-URL VAR_GITLAB_URL]
                       [--var-project-name VAR_PROJECT_NAME]
                       [--var-start-date VAR_START_DATE]
                       [--var-current-status VAR_CURRENT_STATUS]
                       [--var-size VAR_SIZE] [--var-inodes VAR_INODES]
                       [--var-size-follow VAR_SIZE_FOLLOW]
                       [--var-inodes-follow VAR_INODES_FOLLOW]
                       [--var-snakemake-nb VAR_SNAKEMAKE_NB]
                      project filename
```

Positional Arguments

project	Path of project directory
filename	README.md path & filename

Named Arguments

--skip-collect, -s	Skip the collection of file size & inodes Default: False
--is-valid, -t	Test validity of existing README file Default: False
--var-directory	template variable 'directory'
--var-PI-name	template variable 'PI_name'
--var-PI-email	template variable 'PI_email'
--var-archiver-name	template variable 'archiver_name'
--var-archiver-email	template variable 'archiver_email'
--var-CUBI-name	template variable 'CUBI_name'
--var-client-name	template variable 'client_name'
--var-client-email	template variable 'client_email'
--var-SODAR-UUID	template variable 'SODAR_UUID'
--var-SODAR-URL	template variable 'SODAR_URL'
--var-Gitlab-URL	template variable 'Gitlab_URL'
--var-project-name	template variable 'project_name'
--var-start-date	template variable 'start_date'
--var-current-status	template variable 'current_status'
--var-size	template variable 'size'
--var-inodes	template variable 'inodes'
--var-size-follow	template variable 'size_follow'
--var-inodes-follow	template variable 'inodes_follow'
--var-snakemake-nb	template variable 'snakemake_nb'

summary

Collects a summary of files in the project directory. The summary can be saved to a file for further inspection

```
cubi-tk archive summary [-h] [--classes CLASSES] [--dont-follow-links]
                        project table
```

Positional Arguments

project Path of project directory

table Location of the summary output table

Named Arguments

--classes Location of the file describing files of interest

Default: "/home/docs/checkouts/readthedocs.org/user_builds/cubi-tk/checkouts/latest/cubi_tk/archive/classes.yaml"

--dont-follow-links Do not follow symlinks to directories. Required when the project contains circular symlinks

Default: False

CHAPTER 3

Manual for `isa-tpl`

`cubi-tk isa-tpl`: create ISA-tab directories using Cookiecutter.

You can use this command to quickly bootstrap an ISA-tab investigation. The functionality is built on Cookiecutter.

To create a directory with ISA-tab files, run:

```
$ cubi-tk isa-tpl <template name> <output directory>
```

This will prompt a number of questions interactively on the command line to collect information about the files that are going to be created. The requested information will depend on the chosen ISA-tab template. It is also possible to pass this information non-interactively together with other command line arguments (see `cubi-tk isa-tpl <template name> --help`).

The completed information will then be used to create a directory with ISA-tab files. It will be necessary to edit and extend the automatically generated files, e.g. to add additional rows to the assays.

3.1 Available Templates

The Cookiecutter directories are located in this module's directory. Currently available templates are:

- `isatab-generic`
- `isatab-germline`
- `isatab-microarray`
- `isatab-ms_meta_biocrates`
- `isatab-single_cell_rnaseq`
- `isatab-bulk_rnaseq`
- `isatab-tumor_normal_dna`
- `isatab-tumor_normal_triplets`
- `isatab-stem_cell_core_bulk`

- `isatab-stem_cell_core_sc`

3.2 Adding Templates

Adding templates consists of the following steps:

1. Add a new template directory below `cubi_tk/isa_tpl`.
2. Register it appending a `IsaTabTemplate` object to `_TEMPLATES` in `cubi_tk.isa_tpl`.
3. Add it to the list above in the docstring.

The easiest way to start out is to copy an existing cookiecutter template and registration.

3.3 More Information

Also see `cubi-tk isa-tpl` CLI documentation and `cubi-tk isa-tab --help` for more information.

CHAPTER 4

Manual for `isa-tab`

`cubi-tk isa-tab`: ISA-tab tooling.

4.1 Sub Commands

`validate` Validate ISA-tab files for correctness and perform sanity checks.

`resolve-hpo` Resolve lists of HPO terms to TSV suitable for copy-and-paste into ISA-tab.

`add-ped` Given a germline DNA sequencing ISA-tab file and a PED file, add new lines to the ISA-tab file and update existing ones, e.g., for newly added parents.

`annotate` Add annotation to an ISA-tab file, given a tsv file.

4.2 Annotate

`cubi-tk isa-tab annotate` updates material and file nodes in ISA-tab studies and assays with annotations provided as tab-separated text file.

In the annotation file header, target node types need to be indicated in ISA-tab style (i.e. "Source Name", etc.) while annotations are just named normally. Annotations for materials are automatically recorded as Characteristics, while annotations for files are recorded as Comments. Different node types can be annotated using only one annotation file, as demonstrated in the example below.

By default, if Characteristics or Comments with the same name already exist for a node type, only empty values are updated. Overwriting existing values requires confirmation (*–force-update*).

Annotations are only applied to one study and assay, since material names are not necessarily unique between the same material types of different studies or different assays (and thus, annotations couldn't be assigned unambiguously). By default the first study and assay listed in the investigation file are considered for annotation. A specific study and assay may be selected by file name (not path, just as listed in the investigation file) via *–target-study* or *–target-assay*, resp.

Example execution:

```
$ cubi-tk isa-tab annotate investigation.tsv annotation.tsv --target-study s_study.tsv
--target-assay a_assay.tsv
```

Note: investigation.tsv and annotation.tsv have to be indicated via absolute or relative paths. However, s_study.tsv and a_assay.tsv have to be indicated by name only, just as they are referenced in their corresponding investigation file.

Table 1: Annotation example tsv file

Source Name	Age	Sex	Sample Name	Volume
alpha	18	FEMALE	alpha-N1	1000
beta	27	MALE	beta-N1	1000
gamma	69	FEMALE	gamma-N1	800

4.3 More Information

Also see `cubi-tk isa-tab` CLI documentation and `cubi-tk isa-tab --help` for more information.

CHAPTER 5

Manual for `ingest-fastq`

The `cubi-tk sodar ingest-fastq` command lets you upload raw data files to SODAR. It is configured for uploading FASTQ files by default, but the parameters can be adjusted to upload any files.

The basic usage is:

```
$ cubi-tk sodar ingest-fastq SOURCE [SOURCE ...] DESTINATION
```

where each `SOURCE` is a path to a folder containing relevant files and `DESTINATION` is either an iRODS path to a *landing zone* in SODAR or the UUID of that *landing zone*.

5.1 Other file types

By default, the parameters `--src-regex` and `--remote-dir-pattern` are configured for FASTQ files, but they may be changed to upload other files as well. The two parameters have the following functions:

- `--src-regex`: a regular expression to recognize paths to raw data files to upload (the paths starting from the `SOURCE` directories).
- `--remote-dir-pattern`: a pattern specifying into which folder structure the raw data files should be uploaded. This is a file path with wildcards that are replaced by the captured content of named groups in the regular expression passed via `--src-regex`.

For example, the default `--src-regex` is

```
(.*/)?(?P<sample>.+?)(?:_(?P<lane>L[0-9]+?))?(?:_(?P<mate>R[0-9]+?))?(?:_(?P<batch>[0-
↪9]+?))?\.f(?:ast)?q\.gz
```

It can capture a variety of different FASTQ file names and has the named groups `sample`, `lane`, `mate` and `batch`. The default `--remote-dir-pattern` is

```
{sample}/{date}/{filename}
```

It contains the wildcard `{sample}`, which will be filled with the captured content of group `(?P<sample>...)`. In addition, the wildcards `{date}` and `{filename}` can always be used and will be filled with the current date and full filename (the basename of a matched file), respectively.

5.2 Mapping of file names

In some cases additional mapping of filenames is required (for example the samples should be renamed). This can be done via the parameter `--remote-dir-mapping` or short `-m`. It can be supplied several times, each time for another mapping. With each `-m MATCH REPL` a pair of a regular expression and a replacement string are specified. Internally, pythons `re.sub` command is executed on the `--remote-dir-pattern` after wildcards have been filled. Therefore, you can refer to the documentation of the re package for syntax questions.

5.3 Source files on WevDAV

If a `SOURCE` is a WebDAV url, the files will temporarily be downloaded into a directory called "./temp/". This can be adjusted with the `--tmp` option.

5.4 SODAR authentication

To use this command, which internally executes iRODS icommands, you need to authenticate with iRODS by running:

```
$ iinit
```

To be able to access the SODAR API (which is only required, if you specify a landing zone UUID instead of an iRODS path), you also need an API token. For token management for SODAR, the following docs can be used:

- https://sodar.bihealth.org/manual/ui_user_menu.html
- https://sodar.bihealth.org/manual/ui_api_tokens.html

There are three options how to supply the token. Only one is needed. The options are the following:

1. configure `~/.cubitkrc.toml`.

   ```
   [global]

   sodar_server_url = "https://sodar.bihealth.org/"
   sodar_api_token = "<your API token here>"
   ```

2. pass via command line.

   ```
   $ cubi-tk sodar ingest-fastq --sodar-url "https://sodar.bihealth.org/" --
   ↪sodar-api-token "<your API token here>"
   ```

3. set as environment variable.

   ```
   $ SODAR_API_TOKEN="<your API token here>"
   ```

5.5 More Information

Also see `cubi-tk sodar ingest-fastq` *CLI documentation* and `cubi-tk sodar ingest-fastq --help` for more information.

CHAPTER 6

Manual for `sea-snap itransfer-results`

The `cubi-tk sea-snap itransfer-results` command lets you upload results of the Seasnap pipeline to SODAR. It relies on running the `export` function of Seasnap first. This `export` function allows to select which result files of the pipeline shall be uploaded into what folder structure, which can be configured via the Seasnap config file. It outputs a `blueprint` file with file paths and commands to use for the upload. For more information see the Seasnap documentation The `itransfer-results` function parallelizes the upload of these files.

The basic usage is:

1. create blueprint

```
$ ./sea-snap mapping l export
```

2. upload to SODAR

```
$ cubi-tk sea-snap itransfer-results BLUEPRINT DESTINATION
```

where each `BLUEPRINT` is the blueprint file mentioned above (probably "SODAR_export_blueprint.txt") and `DESTINATION` is either an iRODS path to a *landing zone* in SODAR or the UUID of that *landing zone*.

6.1 SODAR authentication

To use this command, which internally executes iRODS icommands, you need to authenticate with iRODS by running:

```
$ iinit
```

To be able to access the SODAR API (which is only required, if you specify a landing zone UUID instead of an iRODS path), you also need an API token. For token management for SODAR, the following docs can be used:

- https://sodar.bihealth.org/manual/ui_user_menu.html
- https://sodar.bihealth.org/manual/ui_api_tokens.html

There are three options how to supply the token. Only one is needed. The options are the following:

1. configure `~/.cubitkrc.toml`.

```
[global]

sodar_server_url = "https://sodar.bihealth.org/"
sodar_api_token = "<your API token here>"
```

2. pass via command line.

```
$ cubi-tk sodar ingest-fastq --sodar-url "https://sodar.bihealth.org/" --
↪sodar-api-token "<your API token here>"
```

3. set as environment variable.

```
$ SODAR_API_TOKEN="<your API token here>"
```

6.2 More Information

Also see `cubi-tk sea-snap itransfer-results` *CLI documentation* and `cubi-tk sea-snap itransfer-results --help` for more information.

CHAPTER 7

Manual for `sea-snap write-sample-info`

The `cubi-tk sea-snap write-sample-info` command can be used to collect information by parsing the folder structure of raw data files (FASTQ) and meta-information (ISA-tab). It collects this information in a YAML file that will be loaded by the Seasnap pipeline.

The basic usage is:

```
$ cubi-tk sea-snap write-sample-info IN_PATH_PATTERN
```

where `IN_PATH_PATTERN` is a file path with wildcards specifying the location to FASTQ files. The wildcards are also used to extract information from the parsed paths.

By default, a file called `sample_info.yaml` will be generated in the current working directory. If this file is in the project working directory, Seasnap will load it automatically. However, you can specify another file name after `IN_PATH_PATTERN`. Then this file can be used in Seasnap e.g. like so:

```
$ ./sea-snap mapping l --config file_name='sample_info_alt.yaml'
```

Note: check and edit the auto-generated sample_info.yaml file before running the pipeline.

7.1 Path pattern and wildcards

For example, if the FASTQ files are stored in a folder structure like this:

```
input
├── sample1
│   ├── sample1_R1.fastq.gz
│   └── sample1_R2.fastq.gz
└── sample2
    ├── sample2_R1.fq
    └── sample2_R2.fq
```

Then the path pattern can look like the following:

```
$ cubi-tk sea-snap write-sample-info "input/{sample}/*_{mate,R1|R2}"
```

Keywords in braces (e.g. `{sample}`) are wildcards. It is possible to add a regular expression separated with a comma after the keyword. This is useful to restrict what part of the file path the wildcard can match (e.g. `{mate,R1|R2}` means that mate can only be `R1` or `R2`). In addition, `*` and `**` can be used to match anything that does not need to be captured with a wildcard.

Setting the `IN_PATH_PATTERN` as shown above will allow the `write-sample-info` command to extract the information that samples *sample1* and *sample2* exist and that there are *paired reads* for both of them. The extension (e.g. `fastq.gz`, `fastq` or `fq`) should be omitted and will be detected automatically.

Available wildcards are: `{sample}`, `{mate}`, `{flowcell}`, `{lane}`, `{batch}` and `{library}`. However, only **"{sample}"** is obligatory.

Note: wildcards do not match "/" and".". For further information also see the Seasnap docu.

7.2 Meta information

When working with **SODAR**, additional meta-information should be included in the sample info file. In SODAR this meta-information is stored in the form of ISA-tab files.

There are two ways to add the information from an ISA-tab assay file to the generated sample info file:

1. Load from a local ISA-tab assay file

```
$ cubi-tk sea-snap write-sample-info --isa-assay PATH/TO/a_FILE_NAME.txt IN_PATH_
↪PATTERN
```

2. Download from SODAR

```
$ cubi-tk sea-snap write-sample-info --project_uuid UUID IN_PATH_PATTERN
```

Here, `UUID` is the UUID of the respective project on SODAR.

7.3 SODAR authentication

To be able to access the SODAR API (which is only required if you download meta-data from SODAR), you also need an API token. For token management for SODAR, the following docs can be used:

- https://sodar.bihealth.org/manual/ui_user_menu.html
- https://sodar.bihealth.org/manual/ui_api_tokens.html

There are three options how to supply the token. Only one is needed. The options are the following:

1. configure `~/.cubitkrc.toml`.

```
[global]

sodar_server_url = "https://sodar.bihealth.org/"
sodar_api_token = "<your API token here>"
```

2. pass via command line.

```
$ cubi-tk sodar ingest-fastq --sodar-url "https://sodar.bihealth.org/" --
↪sodar-api-token "<your API token here>"
```

3. set as environment variable.

```
$ SODAR_API_TOKEN="<your API token here>"
```

7.4 Table format

Although this is not really necessary to run the workflow, it is possible to convert the YAML file to a table / sample sheet:

```
$ cubi-tk sea-snap write-sample-info --from-file sample_info.yaml XXX sample_info.tsv
```

And back:

```
$ cubi-tk sea-snap write-sample-info --from-file sample_info.tsv XXX sample_info.yaml
```

7.5 More Information

Also see `cubi-tk sea-snap write-sample-info` *CLI documentation* and `cubi-tk sea-snap write-sample-info --help` for more information.

CHAPTER 8

Manual for `archive`

The `cubi-tk archive` is designed to facilitate the archival of older projects away from the cluster's fast file system. This document provides an overview of these commands, and how they can be adapted to meet specific needs.

8.1 Glossary

Hot storage: Fast and expensive, therefore usually size restricted. For example:

- GPFS by DDN (currently at `/fast`)
- Ceph with SSDs

Warm storage: Slower, but with more space and possibly mirroring. For example:

- SODAR with irods
- Ceph with HDDs (`/data/cephfs-2/`)

Cold storage: For data that needs to be accessed only rarely. For example:

- Tape archive

8.2 Background: the archiving process

CUBI archive resources are three-fold:

- SODAR and associated irods storage should contain raw data generated for the project. SODAR also contains important results (mapping, variants, differential expression, ...).
- Gitlab contains small files required to generate the results, typically scripts, configuration files, READMEs, meeting notes, ..., but also knock-in gene sequence, list of papers, gene lists, etc.
- The rest should be stored in CEPH (warm storage).

For older projects or intermediate results produced by older pipelines the effort of uploading the data to SODAR & gitlab may not be warranted. In this case, the bulk of the archive might be stored in the CEPH file system.

The module aims to facilitate this last step, i.e. the archival of old projects to move them away from the hot storage.

8.3 Archiving process requirements

Archived projects should contain all **important** files, but not data already stored elsewhere. In particular, the following files should **not** be archived:

- raw data (`*.fastq.gz` files) saved in SODAR or in the `STORE`,
- data from public repositories (SRA, GDC portal, ...) that can easily be downloaded again,
- static data such as genome sequence & annotations, variant databases from gnomAD, ... that can also be easily retrieved,
- indices files for mapping that can be re-generated.

Importantly, a README file should be present in the archive, briefly describing the project, listing contacts to the client & within CUBI and providing links to SODAR & Gitlab when appropriate.

The purpose of the module is:

- to provide a summary of files that require special attention, for example symlinks which targets lie outside of the project, or large files (`*.fastq.gz` or `*.bam` especially)
- to create a temporary directory that mimicks the archived files with symlinks,
- to use this temporary directory as template to copy files on the CEPH filesystem, and
- to compute checksums on the originals and copies, to ensure accuracy of the copy process.

8.4 Basic usage

8.4.1 Summary of files in project

```
$ cubi-tk archive summary PROJECT_DIRECTORY DESTINATION
```

Unlike other `cubi-tk` commands, here `DESTINATION` is not a landing zone, but a local filename for the summary of files that require attention.

By default, the summary reports:

- dangling symlinks (also dangling because of permission),
- symlinks pointing outside of the project directory,
- large (greater than 256MB) `*.fastq.gz`, `*.fq.gz` & `*.bam` files,
- large static data files with extension `*.gtf`, `*.gff`, `*.fasta` & `*.fa` (possibly gzipped), that can potentially be publicly available.
- large files from SRA with prefix `SRR`.

The summary file is a table with the following columns:

- **Class**: the name(s) of the pattern(s) that match the file. When the file matches several patterns, all are listed, separated by `|`.

- **Filename**: the relative path of the file (from the project's root).
- **Target**: the symlink's target (when applicable)
- **ResolvedName**: the resolved (absolute, symlinks removed) path of the target. When the target doesn't exist or is inaccessible because of permissions, the likely path of the target.
- **Size**: file size (target file size for symlinks). When the file doesn't exist, it is set to 0.
- **Dangling**: `True` when the file cannot be read (missing or inaccessible), `False` otherwise.
- **Outside**: `True` when the target path is outside of the project directory, `False` otherwise. It is always `False` for real files (_i.e._ not symlinks).

The summary step also reports an overview of the results, with the total number of files, the total size of the project, and the number of links to files. Number of dangling links and links inaccessible because of permission issues are listed separately. Likewise, the number of files outside of the projects, which are linked to from within the project by symlinks is also quoted. Finally, for each of the "important files" classes, the number of files, the number of files outside of the project directory and the number of files lost because of symlink failures are reported.

8.4.2 Archive preparation: README.md file creation

```
$ cubi-tk archive readme PROJECT_DIRECTORY README_FILE
```

`README_FILE` is here the path to the README file that will be created. It must not exist.

The README file will be created by filling contact information interactively. Command-line options are also available, but interactive confirmation is needed.

It is possible to test if a generated README file is valid for project archival, using

```
$ cubi-tk archive readme --is-valid PROJECT_DIRECTORY README_FILE
```

The module will highlight mandatory records that could not be found in the current file. These mandatory records are lines following the patterns below:

```
- P.I.: [Name of the PI, any string](mailto:<valid email address in lowercase>)
- Client contact: [Name of our contact in the PI's group](mailto:<valid email address␣
↪in lowercase>)
- CUBI project leader: [Name of the CUBI member leading the project]
- CUBI contact: [Name of the archiver](mailto:<valid email address in lowercase>)
- Project name: <any string>
- Start date: YYYY-MM-DD
- Current status: <One of Active, Inactive, Finished, Archived>
```

8.4.3 Archive preparation: temporary copy

```
$ cubi-tk archive prepare --readme README PROJECT_DIRECTORY TEMPORARY_DESTINATION
```

`TEMPORARY_DESTINATION` is here the path to the temporary directory that will be created. It must not exist.

For each file that must be archived, the module creates a symlink to that file's absolute path. The module also reproduces the project's directories hierarchy, so that the symlink sits in the same relative position in the temporary directory than in the original project.

The module deals with symlinks in the project differently whether their target in inside the project or not. For symlinks pointing outside of the project, a symlink to the target's absolute path is created. For symlinks pointing inside the

project, a relative path symlink is created. This allows to store all files (even those outside of the project), without duplicating symlinks inside the project.

Additional transformation of the original files are carried out during the preparation step:

- The contents of the `.snakemake`, `sge_log`, `cubi-wrappers` & `snappy-pipeline` directories are processed differently: the directories are tarred & compressed in the temporary destination, to reduce the number of inodes in the archive.
- The core dump files are not copied to the temporary destination, and therefore won't be copied to the final archive.
- The `README.md` file created by the `readme` subcommand must also be included to be put in the temprary's destination top level. If the original project already contains a `README.md` file, it will be appended to the generated one, as the latter is valid (it contains all mandatory information).

8.4.4 Copy to archive & verification

```
$ cubi-tk archive copy TEMPORARY_DESTINATION FINAL_DESTINATION
```

`FINAL_DESTINATION` is here the path to the final destination of the archive, on the warm storage. It must not exist.

8.5 Configuration

The files reported in the summary are under user control, through the `--classes` option, which must point to a yaml file describing the regular expression pattern & minimum size for each class. For example, raw data files can be identified as follows:

```
fastq:
    min_size: 268435456
    pattern: "^(.*/)?[^/]+(\\.f(ast)?q(\\.gz)?)$"
```

The files larger than 256MB, with extension `*.fastq`, `*.fq`, `*.fastq.gz` or `*.fq.gz` will be reported with the class `fastq`. Any number of file class can be defined. The default classes configuration is in `cubi_tk/archive/classes.yaml`

The behaviour of the archive preparation can also be changed using the `--rules` option. The rules are also described in a yaml file by regular expression patterns.

Three different archiving options are implemented:

- **ignore**: the files or directories matching the pattern are simply omitted from the temporary destination. This is useful to ignore remaining temporary files, core dumps or directories containing lists of input symlinks, for example.
- **compress**: the files or directories matching the pattern will be replaced in the temporary destination by a compressed (gzipped) tar file. This is how `.snakemake` or `sge_log` directories are treated by default, but patterns for other directories may be added, for example for the Slurm log directories.
- **squash**: the files matching the pattern will be replaced by zero-length placeholders in the temporary destination. A md5 checksum file will be added next to the original file, to enable verification.

When the user doesn't specify her own set using the `--rules` option, the rules applied are the following: core dumps are ignored, `.snakemake`, `sge_log`, `.git`, `snappy-pipeline` and `cubi_wrappers` directories are compressed, and nothing is squashed. The exact definitions are:

```
ignore:             # Patterns for files or directories to skip
    - "^(.*/)?core\\.[0-9]+$"
    - "^(.*/)?\\.venv$"

compress:           # Patterns for files or directories to tar-gzip
    - "^(.*/)?\\.snakemake$"
    - "^(.*/)?sge_log$"
    - "^(.*/)?\\.git$"
    - "^(.*/)?snappy-pipeline$"
    - "^(.*/)?cubi_wrappers$"

squash: []          # Patterns for files to squash (compute MD5 checksum, and replace␣
↪by zero-length placeholder)
```

8.6 Examples

Consider an example project. It contains:

- raw data in a `raw_data` directory, some of which is stored outside of the project's directory,
- processing results in the `pipeline` directory,
- additional data files & scripts in `extra_data`,
- a `.snakemake` directory that can potentially contain many files in conda environments, for example, and
- a bunch on temporary & obsolete files that shouldn't be archived, conveniently grouped into the `ignored_dir` directory.

The architecture of this toy project is displayed below:

```
project/
├── extra_data
│   ├── dangling_symlink -> ../../outside/inexistent_data
│   ├── file.public
│   ├── to_ignored_dir -> ../ignored_dir
│   └── to_ignored_file -> ../ignored_dir/ignored_file
├── ignored_dir
│   └── ignored_file
├── pipeline
│   ├── output
│   │   ├── sample1
│   │   │   └── results -> ../../work/sample1/results
│   │   └── sample2 -> ../work/sample2
│   └── work
│       ├── sample1
│       │   └── results
│       └── sample2
│           └── results
├── raw_data
│   ├── batch1 -> ../../outside/batch1
│   ├── batch2
│   │   ├── sample2.fastq.gz -> ../../../outside/batch2/sample2.fastq.gz
│   │   └── sample2.fastq.gz.md5 -> ../../../outside/batch2/sample2.fastq.gz.md5
│   └── batch3
│       ├── sample3.fastq.gz
│       └── sample3.fastq.gz.md5
│
```

(continues on next page)

(continued from previous page)

```
└── .snakemake
    └── snakemake
```

8.6.1 Prepare the copy on the temporary destination

Imagine now that the raw data is already safely archived in SODAR. We don't want to save these files in duplicate, so we decide ito _squash_ the raw data files so that their size is set to 0, and their md5 checksum is added. We also do the same for the publicly downloadable file `file.public`. We also want to ignore the junk in `ignored_dir`, and to compress the `.snakemake` directory. So we have the following rules:

After running the preparation command `cubi-tk archive prepare --rules my_rules.yaml project temp_dest`, the temporary destination contains the following files:

```
temp_dest
├── <today's date>_hashdeep_report.txt
├── extra_data
│   ├── file.public
│   ├── file.public.md5
│   ├── to_ignored_dir -> ../ignored_dir
│   └── to_ignored_file -> ../ignored_dir/ignored_file
├── pipeline
│   ├── output
│   │   ├── sample1
│   │   │   └── results -> ../../work/sample1/results
│   │   └── sample2 -> ../work/sample2
│   └── work
│       ├── sample1
│       │   └── results -> /absolute_path/project/pipeline/work/sample1/results
│       └── sample2
│           └── results -> /absolute_path/project/pipeline/work/sample2/results
├── raw_data
│   ├── batch1
│   │   ├── sample1.fastq.gz
│   │   └── sample1.fastq.gz.md5 -> /absolute_path/outside/batch1/sample1.fastq.gz.md5
│   ├── batch2
│   │   ├── sample2.fastq.gz
│   │   └── sample2.fastq.gz.md5 -> /absolute_path/outside/batch2/sample2.fastq.gz.md5
│   └── batch3
│       ├── sample3.fastq.gz
│       └── sample3.fastq.gz.md5 -> /absolute_path/project/raw_data/batch3/sample3.
↪fastq.gz.md5
├── README.md
└── .snakemake.tar.gz
```

The inaccessible file `project/extra_data/dangling_symlink` & the contents of the `project/ignored_dir` are not present in the temporary destination, either because they are not accessible, or because they have been conscientiously ignored by the preparation step.

The `.snakemake` directory is replaced by the the gzipped tar file `.snakemake.tar.gz` in the temporary destination.

The `file.public` & the 3 `*.fastq.gz` files have been replaced by placeholder files of size 0. For `file.public`, the md5 checksum has been computed by the preparing step, but for the `*.fastq.gz` files, the existing checksums are used.

All other files are kept for archiving: symlinks for real files point to their target's absolute path, symlinks are absolute for paths outside of the project, and relative for paths inside the project.

Finally, the hashdeep report of the original project directory is written to the temporary destination, and a `README.md` file is created. **At this point, we edit the ''README.md'' file to add a meaningful description of the project.** If a `README.md` file was already present in the orginial project directory, its content will be added to the newly created file.

Note that the symlinks `temp_dest/extra_data/to_ignored_dir` & `temp_dest/extra_data/to_ignored_file` are dangling, because the link themselves were not omitted, but their targets were. **This is the expected, but perhaps unwanted behaviour**: symlinks pointing to files or directories within compressed or ignored directories will be dangling in the temporary destination, as the original file exists, but is not part of the temporary destination.

8.6.2 Copy to the final destination

When the `README.md` editing is complete, the copy to the final destination on the warm file system can be done. It is matter of `cubi-tk archive copy temp_dest final_dest`.

The copy step writes in the final destination the hashdeep audit of the copy against the original project. This audit is expected to fail, because files & directories are ignored, compressed or squashed. The option `--keep-workdir--hashdeep`, the programme also outputs the hashdeep report of the temporary destination, and the audit of the final copy against the temporary destination. Both the report and the audit are also stored in the final copy directory. The audit of the copy against the temporary destination should be successful, as the copy doesn't re-process files, it only follows symlinks.

If all steps have been completed successfully (including checking the `README.md` for validity), then a marker file named `archive_copy_complete` is created. The final step is to remove write permissions if the `--read-only` option was selected.

8.7 Additional notes and caveats

- Generally, the module doesn't like circular symlinks. It is wise to fix them before any operation, or use the rules facility to ignore them during preparation. The `--dont-follow-links` option in the summary step prevents against such problems, at the expense of missing some files in the report.
- The module is untested for symlink corner cases (for example, where a symlink points to a symlink outside of the project, which in turn points to another file in the project).
- In the archive, relative symlinks within the project are resolved. For example, in the original project one might have `variants.vcf -> ../work/variants.vcf -> variants.somatic.vcf`. In the archive, the link will be `variants.vcf -> ../work/variants.somatic.vcf`.

8.8 More Information

Also see `cubi-tk archive --help`, `cubi-tk archive summary --help`, `cubi-tk archive prepare --help` & `cubi-tk archive copy --help` for more information.

CHAPTER 9

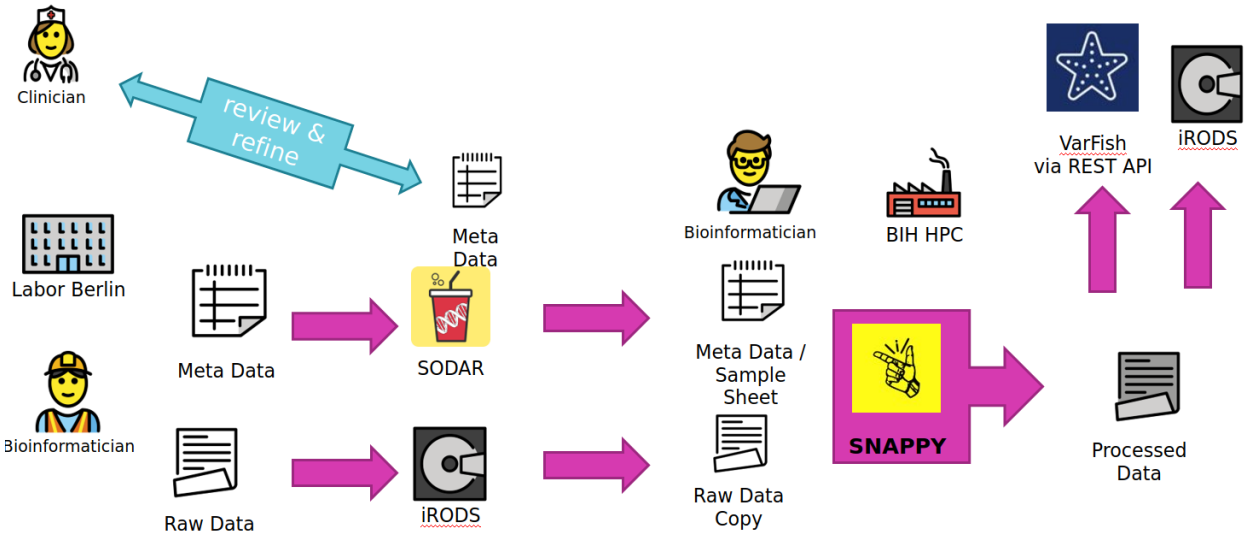
Use Case: Exomes

This section describes the cubi-tk use case for exomes that are sequenced at Labor Berlin and processed by CUBI. This section provides an outline of how cubi-tk helps in connecting

- SODAR (the CUBI system for meta and mass data storage and management),
- SNAPPY (the CUBI pipeline for the processing of DNA sequencing, including exomes),
- and VarFish (the CUBI web app for interactive analysis and annotation of variant calling results).

9.1 Overview

The overall data flow for the Translate-NAMSE use case is depicted below.



- A Labor Berlin (LB) bioinformatician uses "cubi-tk sodar add-ped" to augment the sample sheet of a SODAR project with new family members or new families alltogether. He also transfers the FASTQ read data sequences to the iRODS system that backs SODAR for file storage.
- At this stage, a Charite geneticist can review and refine the sample sheet. This mostly relates to information that is secondary for the subsequent analysis. It is assumed that the family relations updated by the bioinformatician are correct (two parents of a sample are the two parents, if father and mother are flipped, this is not important for analysis by SNAPPY).
- A CUBI Bioinformatician can now update the sample sheet for the SNAPPY pipeline using "cubi-tk snappy pull-sheets" and update a copy of the raw data sequence with "cubi-tk snappy pull-raw-data" files earlier transferred by LB.
- Once the data has been pulled from SODAR and iRODS, the CUBI bioinformatician launches the SNAPPY pipeline which processes the data on the BIH HPC. The command `cubi-tk snappy kickoff` launches the pipeline steps with their dependencies. Inspection of results is based on manual inspection of log files for now.
- Once this is complete, Manuel uses `cubi-tk snappy varfish-upload` and `cubi-tk snappy itransfer-{variant-calling,ngs-mapping}` to transfer the resulting BAM and VCF files into VarFish via its REST API and iRODS via landing zones (`cubi-tk sodar lz-{create,move}`).

To summarise more concisely

- LB copies data and meta data to SODAR/iRODS.
- CUBI pulls mass data and meta data form SODAR/iRODS and starts the pipeline.
- CUBI submits the resulting mass data results back into SODAR and annotated/exported variant calls into VarFish.
- The clinician can review the sample sheet independently of Manuel and Johannes.

Human interaction is required if

- The sample sheet does not sufficiently reflect reality (sample swaps)
- Files are broken and/or swapped.
- Tools terminate too early; data is not copied.
- Overall, this is not fully automated system, rather a system with heavy tool support and semi-automation.

Future improvements are

- Ask clinicians sending in samples for sex of child.
- Properly track parents as father/mother.

More Notes

- Data is processed in batches.
- Many tooling steps rely on "start processing in batch NUMBER"
- That is, everything behind NUMBER will be processed.
- Requires human-manual tracking of batch to start at (easy to seee in SODAR)

9.2 Setup

For token management for both VarFish and SODAR, the following docs can be used:

- https://sodar.bihealth.org/manual/ui_user_menu.html

- https://sodar.bihealth.org/manual/ui_api_tokens.html

1. Obtain a VarFish API token from the varfish system and configure `~/.varfishrc.toml`.

```
[global]
varfish_server_url = "https://varfish.bihealth.org/"
varfish_api_token = "<your API token here>"
```

2. Obtain a SODAR API token and configure `~/.cubitkrc.toml`.

```
[global]

sodar_server_url = "https://sodar.bihealth.org/"
sodar_api_token = "<your API token here>"
```

3. Create a new Miniconda installation if necessary.

```
host:~$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
↪x86_64.sh
host:~$ bash Miniconda3-latest-Linux-x86_64.sh -b -p $HOME/miniconda3
host:~$ source $HOME/miniconda3/bin/activate
(conda) host:~$
```

4. Checkout and install VarFish CLI:

```
(conda) host:~$ git clone https://github.com/bihealth/varfish-cli.git
(conda) host:~$ cd varfish-cli
(conda) host:varfish-cli$ pip install -r requirements/base.txt
(conda) host:varfish-cli$ pip install -e .
```

5. Checkout and install CUBI-TK

```
(conda) host:~$ git clone git@cubi-gitlab.bihealth.org:CUBI/Pipelines/
↪cubi-tk.git
(conda) host:~$ cd cubi-tk
(conda) host:cubi-tk$ pip install -r requirements/base.txt
(conda) host:cubi-tk$ pip install -e .
```

9.3 SNAPPY Configuration

You have to adjust the configuration of the SNAPPY data set as follows:

- You have ot provide the `sodar_uuid` attribute. Set it to the SODAR project's UUID.
- **Data will be downloaded in the last entry of `search_paths`.**
  - If you are starting a new project then just use one entry with an appropriate value.
  - If you are moving a project to use cubi-tk then add a new entry where to download the data to.

```
# ...
data_sets:
  "<the dataset name here>:
    sodar_uuid: "<dataset uuid here>
    sodar_title: "<optional title here>
    file: "<biomedsheets file path here>.tsv"
    type: germline_variants
```

(continues on next page)

(continued from previous page)

```
    naming_scheme: only_secondary_id
    search_patterns:
    - {left: '**/*_R1.fastq.gz', right: '**/*_R2.fastq.gz'}
    - {left: '**/*_R1_*.fastq.gz', right: '**/*_R2_*.fastq.gz'}
    search_paths:
    - "<path to search data for here>"
```

Note that you will need the **/* in the pattern.

9.4 Processing Commands

The setup up to here only has to be done only once for each project/dataset. The following step will (a) fetch the meta data and raw data from SODAR/iRODS, (b) start the processing with SNAPPY, and (c) submit the results back to SODAR once SNAPPY is done.

First, you pull the meta data from SODAR with the command:

```
$ cubi-tk snappy pull-sheets
```

This will show the changes that are to be applied in unified patch format and you have to confirm by files. You can also add `--yes --dry-run` to see all pending changes at once without actually applying them or `--yes` to apply all changes.

The next step is to fetch the raw data from SODAR/iRODS. You first have to authenticate with iRODS using `init`. You then fetch the raw data, optionally only the data starting at batch number `$BATCH`. You also have to provide the project UUID `$PROJECT`. Internally, cubi-tk will use the iRODS icommands and you will be shown the commands it is about to execute.

```
$ iinit
$ cubitk snappy pull-raw-data --min-batch $BATCH $PROJECT
```

Now you could start the processing. However, it is advisable to ensure that the input FASTQ files can be linked in the `ngs_mapping` step.

```
$ cd ngs_mapping
$ snappy-snake -p $(snappy-snake -S | grep -v 'no update' | grep input_links | cut -f␣
↪1)
```

If this fails, a good starting point is removing `ngs_mapping/.snappy_path_cache`.

You can kick off the current pipeline using

```
$ cubi-tk snappy kickoff
```

After the pipeline has finished, you can create a new landing zone with the following command. This will print the landing zone properties as JSON. You will neded both the landing zone UUID (`ZONE`) and iRODS path (`$IRODS_PATH`) for now (in the future this will be simplified).

```
$ cubi-tk sodar landing-zone-create $PROJECT
```

You can then transfer the data using the following commands. You will have to specify the path to the SNAPPY sample sheet TSV as `$TSV` and the landing zone iRODS path `$IRODS_PATH`.

```
$ cubi-tk snappy itransfer-ngs-mapping --start-batch $BATCH $TSV $IRODS_PATH
$ cubi-tk snappy itransfer-variant-calling --start-batch $BATCH $TSV $IRODS_PATH
```

Finally, you can validate and move the landing zone to get the data into SODAR:

```
$ cubi-tk sodar landing-zone-move $ZONE
```

And last but not least, here is how to transfer the data into VarFish (starting at `$BATCH`).

```
$ cubi-tk snappy varfish-upload --min-batch $BATCH $PROJECT
```

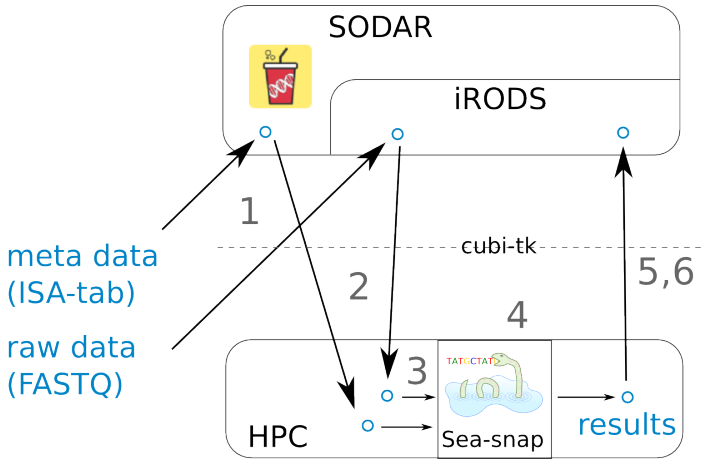
CHAPTER 10

Use Case: Single Cell

This section describes the cubi-tk use case for the analysis of single cell data. It provides an outline of how cubi-tk helps in connecting

- Sea-Snap (the CUBI pipeline for the processing of RNA sequencing, including scRNA-seq),
- SODAR (the CUBI system for meta and mass data storage and management).

10.1 Overview



1 FASTQ and ISA-tab files are uploaded to SODAR.

- ISA-tab files can be created with the help of `cubi-tk isa-tpl isatab-single_cell`.
- FASTQ files can be uploaded with the help of `cubi-tk sodar ingest-fastq`

2 FASTQ and ISA-tab files are pulled from SODAR.

- FASTQ files can be downloaded using `cubi-tk sodar pull-raw-data` or iRods icommands.

- ISA-tab files can be downloaded using `cubi-tk sea-snap pull-isa`.

3 A results folder is created on the HPC cluster and the config files are edited. A sample info file is created.

- A results folder can be created with `cubi-tk sea-snap working-dir`.
- The sample_info.yaml file can be created with `cubi-tk sea-snap write-sample-info`. This combines information from the parsed FASTQ folder structure and ISA-tab meta information.

4 Running the Sea-snap pipeline.

- This is done as usual via `./sea-snap sc --slurm c`.

5 The results are uploaded to SODAR.

- Create a landing zone on SODAR with `cubi-tk sodar lz-create`.
- Create a blueprint of which files to upload with `./sea-snap sc l export`.
- Upload the results using the blueprint and `cubi-tk itransfer-results`.

6 Check whether all files have been uploaded to SODAR correctly.

- This can be done via `cubi-tk sea-snap check-irods`.

10.2 Setup

For token management for SODAR, the following docs can be used:

- https://sodar.bihealth.org/manual/ui_user_menu.html
- https://sodar.bihealth.org/manual/ui_api_tokens.html

1. Obtain a SODAR API token and configure `~/.cubitkrc.toml`.

```
[global]

sodar_server_url = "https://sodar.bihealth.org/"
sodar_api_token = "<your API token here>"
```

2. Create a new Miniconda installation if necessary.

```
host:~$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
↪x86_64.sh
host:~$ bash Miniconda3-latest-Linux-x86_64.sh -b -p $HOME/miniconda3
host:~$ source $HOME/miniconda3/bin/activate
(conda) host:~$
```

3. Checkout and install CUBI-TK

```
(conda) host:~$ git clone git@cubi-gitlab.bihealth.org:CUBI/Pipelines/
↪cubi-tk.git
(conda) host:~$ cd cubi-tk
(conda) host:cubi-tk$ pip install -r requirements/base.txt
(conda) host:cubi-tk$ pip install -e .
```

10.3 Processing Commands

Hint: Also see the Seasnap single cell pipeline documentation here.

First, you can pull the meta data from SODAR with the command:

```
$ cubi-tk sea-snap pull-isa <project_uuid>
```

This will create a folder with ISA-tab files. Alternatively, you can omit this step and automatically pull the files later.

The next step is to fetch the raw data from SODAR/iRODS. You first have to authenticate with iRODS using `iinit`. Internally, cubi-tk will use the iRODS icommands and you will be shown the commands it is about to execute.

```
$ iinit
$ cubi-tk sodar pull-raw-data <project_uuid>
```

Create a working directory for the project results:

```
$ cubi-tk sea-snap working-dir <path_to_seasnap_pipeline>
```

This will also copy relevant files and a config template into the new directory. Edit the config files to adjust the pipeline execution to your needs.

Create a sample info file. This is equivalent to a sample sheet and summarizes information about the samples in yaml format. A path pattern to the downloaded FASTQ files is needed, see Sea-snap doku: https://cubi-gitlab.bihealth.org/CUBI/Pipelines/sea-snap/blob/master/documentation/prepare_input.md#fastq-files-folder-structure

```
$ cubi-tk sea-snap write-sample-info --isa-assay <path_to_assay_file> <path_pattern_
↪to_fastq>
```

This combines information from both the FASTQ folder structure (given via path pattern) and the ISA-tab meta data (given via ISA-assay file). If ISA-tab files have not been downloaded yet, you can use the option `--project-uuid <project_uuid>` instead of `--isa-assay` to download them on-the-fly.

Now you can start the processing. Run the Sea-snap pipeline as usual:

```
$ ./sea-snap sc --slurm c <any snakemake options>
$ ./sea-snap sc --slurm c export
```

After the pipeline has finished, you can create a new landing zone with the following command. This will print the landing zone properties as JSON. You will need the landing zone UUID (`ZONE`) in the next step.

```
$ cubi-tk sodar landing-zone-create <project_uuid>
```

You can then transfer the data using the following commands. You will have to specify the blueprint file generated by the export rule of sea-snap.

```
$ cubi-tk sea-snap itransfer-results <blueprint_file> <landing_zone_uuid>
```

Finally, you can validate and move the landing zone to get the data into SODAR:

```
$ cubi-tk sodar landing-zone-move <landing_zone_uuid>
```

You may check, whether everything was uploaded correctly using the following command:

```
$ cubi-tk sea-snap check-irods <path_to_local_results_folder> <irods_path_to_results_
↪on_sodar>
```

CHAPTER 11

Use Case: Archiving a project

This section describes the process of archiving a project using `cubi-tk`. This section provides an example of how cubi-tk can be used in different cases.

11.1 Overview

The general process to archive projects is:

1. Get acquainted with the contents of the projects directory. The command `cubi-tk archive summary` provides a basic facility to identify several important aspects for the archival process. It does not, however, check whether files are already stored on SODAR. This must be done independently.
2. Archives **must** be accompanied by a `README.md` file, which provides important contact information about the project's scientific P.I., e-mail addresses of the post-doc in charge, the individuals in CUBI that processed the data, and the person in charge of the archive. URLs for SODAR & Gitlab are also important. The command `cubi-tk archive readme` creates a valid README file, that contains these informations.
3. In many cases, not all files should be archived: there is no need to duplicate large sequencing files (fastrq or bam) if they are already safely stored on SODAR. Likewise, whole genome sequence, annotations, indices, should not be archived in most cases. The command `cubi-tk archive prepare` identifies files that must be copied, and those which shouldn't. (it can do a bit more, see below).
4. Once these preparation steps have been carried out, the command `cubi-tk archive copy` performs the copy of the project to its final archive destination. This command creates checksums for all files in the project, and in the archive copy. It provides an audit of the comparison between these two sets of checksums, to ensure trhat the archival was successful.

Each of these steps descibed above are discussed below, to give practical examples, and to suggest good practice.

11.2 Summary

The summarisation step aims to report several cases of files that may require attention for archiving. In particular, symbolic links to destinations outside of the project's directory should be reported. Dangling symbolic links (either because the target is missing, or because of permissions) are also listed.

The module also lists specific files of interest. By default, large bam or fastq files (larger than 256MB) are reported, as well as large fasta files, annotations (with `.gtf` or `.gff` extensions), and short-read-archive sequencing data.

It is possible for the user to change the reporting criteria, using a `yaml` file & the `--classes` option. For example:

```
$ cubi-tk archive summary \
    --classes reporting_classes.yaml \  # Use your own reporting selection
    <project_directory> \
    <summary file>
```

The default summary classes can be found in `<cubi-tk installation>/cubi_tk/archive/classes.yaml`. Its content reads:

```
fastq:
    min_size: 268435456
    pattern: "^(.*/)?[^/]+(\\.f(ast)?q(\\.gz)?)$"
bam:
    min_size: 268435456
    pattern: "^(.*/)?[^/]+(\\.bam(\\.bai)?)$"
public:
    min_size: 268435456
    pattern: "^(.*/)?(SRR[0-9]+[^/]*|[^/]+\\.(fa(sta)?|gtf|gff[23]?)(\\.gz)?)
↪$"
```

The output of the summarization is a table, with the reason why the file is reported in the first column, the file name, the symlink target if the file is a symlink, the file's normalised path, its size, and, in case of symlinks, if the target is accessible, and if it is inside the project or not.

11.3 Readme file creation

The module creates README files that **must** contain contact information to

- The project's scientific P.I. (Name & email address),
- The contact to the person in charge of the project, very often a post-doc in the P.I.'s group (name & e-mail address),
- The contact to the person who is archiving the project (name & e-mail address). This person will be the project's contact in CUBI.
- The name of the person who actually did the data processing & analysis in CUBI. It is generally the same person who is archiving the project, unless he or she has left CUBI.

The SODAR & Gitlab's URLs should also be present in the README file, when applicable. But this information is not mandatory, unlike the contact information.

Important notes

The creation of the README file is a frequent source of errors and frustrations. To minimize the inconveniences, please heed these wise words.

- E-mail addresses must be present, valid & cannot contain uppercase letters (don't ask why...)

- Generally, the module is quite fussy about the format. Spaces, justification, ... may be important.
- Upon README creation, the project directory is quickly scanned to generate an overview of the project's size and number of inodes. For large projects, it is possible to disable this behaviour using the `--skip-collect` option.
- Because of these problems, the module offers a possibility to check README file validity. The command is *cubi-tk archive readme –is-valid project_dir readme_file*.
- If a README file is already present in the project, it will be appended at the bottom of the README file generated by the module.

Most importantly, please edit your README file after generation by the module. The module generates no description of the aims & results of the project, even though it is very useful and important to have.

11.4 Preparation of the copy

During preparation, the user can select the files that will be archived, those that will be discarded, and those that must be processed differently.

The file selection is achieved by creating a temporary copy of the project's directory structure, using symbolic links. The location of this temporary copy is called *temporary destination*.

When copying a file to this temporary destination, its fate is decided based on its filename & path, using regular expression pattern matching. There are 4 types of operations:

- The files are selected for copy. This is the default behaviour.
- Files can be omitted (or *ignored*) from the copy.
- Directories with many (smallish) files can be tarred & compressed to reduce the total number of inodes (which is very file-system friendly).
- Finally, files can be *squashed*. In this case, a file will have its md5 checksum computed and seved in a companion files next to it, and the file will finally be replaced with a placeholder with the same name, but with a size of 0. This is useful for large files that can easily be downloaded again from the internet. Public sequencing datasets, genome sequences & annotations are typical examples.

The user can impose its own rules, based on the content of the project. The selection rules are defined in a yaml file accessed through the module's `--rules` option. The default rules file is in `<cubi-tk installation>/cubi_tk/archive/default_rules.yaml`, and its content reads:

```
ignore:             # Patterns for files or directories to skip
    - "^(.*/)?core\\.[0-9]+$"    # Ignore core dumps
    - "^(.*/)?\\.venv$"          # Ignore virtual environment .venv␣
↪directories

compress:           # Patterns for files or directories to tar-gzip
    - "^(.*/)?\\.snakemake$"     # Created by snakemake process
    - "^(.*/)?sge_log$"          # Snappy SGE log directories
    - "^(.*/)?\\.git$"           # Git internals
    - "^(.*/)?snappy-pipeline$" # Copy of snappy
    - "^(.*/)?cubi_wrappers$"    # Copy of snappy's ancestor

squash: []          # Patterns for files to squash (compute MD5 checksum, and␣
↪replace by zero-length placeholder)
```

Important notes

- The temporary destination is typically chosen as `/fast/scratch/users/<user>/Archive/<project_name>`.
- The README file generated in the previous step is copied to the temporary destination using the module's `--readme` option.
- When the temporary destination is complete, the module creates a complete list of all files accessible from the original project directory, and computes md5 & sh256 checksums, using `hashdeep`. This is done **for all files accessible from the project's directory**, including all symbolic links.
- The computation of checksums can be extremely time-consuming. Multiple threads can be used with the `--num-threads` option. Nevertheless, in most cases, it is advisable to submit the preparation as a slurm job, rather than interactively.

Example of usage:

```
$ cubi-tk archive prepare \
    --rules <my_rules>    \        # Project-specific rules
    --readme <my_readme>  \        # README.md file generated in the previous step
    --ignore-tar-errors   \        # Useful only in cases of inaccessible files to␣
↪compress
    <project_dir>         \
    <termporary_destination>
```

11.5 Copy to final destination

The last step consist in copying all files in the temporary destination to the archiving location. This is done internally using `rsync`, having previously removed all symbolic links connecting files wihtin the project directory. These *local* symbolic links are restored after the copy is complete, in both the temporary & final destinations. After the copy is complete, the archiving directory can be protected against writing with the `--read-only` option.

A verification based on md5 checksums is automatically done between the original project directory and the final copy. In most cases, differences between the directories are expected, because of the files ignored, compressed and squashed. However, it is good practice to examine the audit file to make sure that all files missing from the copy are missing for the right reasons. The report of checksums of all files in the original project, and the audit result are both present in the final destination, as files called `<date>_hashdeep_report.txt` and `<date>_hashdeep_audit.txt` respectively.

For additional verification, it is also possible to request (using the `--keep-workdir-hashdeep` option) a hashdeep report of the temporary destination, and the corresponding audit of the final copy. These contents of these two directories are expected to be identical, and any discrepancy should be looked at carefully. The report & audit files relative to the temporary destination are called `<date>_workdir_report.txt` & `<date>_workdir_audit.txt`.

Finally, the copy and hasdeep steps are quite time-consuming, and it is good practice to submit the copy as a slurm job rather than interactively, even when multiple threads are used (through the `--num-threads` option).

An example of a copy script that can be submitted to slurm is:

```
#!/bin/bash

#SBATCH --job-name=copy
#SBATCH --output=slurm_log/copy.%j.out
#SBATCH --error=slurm_log/copy.%j.err
#SBATCH --partition=medium
#SBATCH --mem=4000
#SBATCH --time=72:00:00
```

(continues on next page)

(continued from previous page)

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8

# ------------------ Command-line options -----------------------------

# Taken from https://stackoverflow.com/questions/402377/using-getopts-to-process-long-
↪and-short-command-line-options
TEMP=$(getopt -o ts:d: --long dryrun,source:,destination: -- "$@")

if [ $? != 0 ] ; then echo "Terminating..." >&2 ; exit 1 ; fi

# Note the quotes around '$TEMP': they are essential!
eval set -- "$TEMP"

dryrun=0
src=""
dest=""
while true; do
    case "$1" in
        -t | --dryrun ) dryrun=1; shift ;;
        -s | --source ) src="$2"; shift 2 ;;
        -d | --destination ) dest="$2"; shift 2 ;;
        -- ) shift; break ;;
        * ) break ;;
    esac
done

if [[ "X$src" == "X" ]] ; then echo "No project directory defined" >&2 ; exit 1 ; fi
if [[ ! -d "$src" ]] ; then echo "Can't find project directory $src" >&2 ; exit 1 ; fi
if [[ "X$dest" == "X" ]] ; then echo "No temporary directory defined" >&2 ; exit 1 ;␣
↪fi
if [[ -e "$dest" ]] ; then echo "Temporary directory $dest already exists" >&2 ; exit␣
↪1 ; fi

if [[ dryrun -eq 1 ]] ; then
    echo "cubi-tk archive copy "
    echo "--read-only --keep-workdir-hashdeep --num-threads 8 "
    echo "\"$src\" \"$dest\""
    exit 0
fi

# ---------------------- Subtmit to slurm -----------------------------

export LC_ALL=en_US
unset DRMAA_LIBRARY_PATH

test -z "${SLURM_JOB_ID}" && SLURM_JOB_ID=$(date +%Y-%m-%d_%H-%M)
mkdir -p slurm_log/${SLURM_JOB_ID}

CONDA_PATH=$HOME/work/miniconda3
set +euo pipefail
conda deactivate &>/dev/null || true  # disable any existing
source $CONDA_PATH/etc/profile.d/conda.sh
conda activate cubi_tk # enable found
set -euo pipefail

cubi-tk archive copy \
```

(continues on next page)

(continued from previous page)

```
    --read-only --keep-workdir-hashdeep --num-threads 8 \
    "$src" "$dest"
```

CHAPTER 12

Credits

- Eudes Barbosa
- Johannes Helmuth
- Manuel Holtgrewe
- Patrick Pett

CHAPTER 13

HISTORY

CHAPTER 14

License

You can find the License of AltamISA below.

```
MIT License

Copyright (c) 2020-2021, Berlin Institute of Health

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Python Module Index

C

Index

C