CAMAMED manual

CAMAMED: a pipeline for composition-aware mapping-based analysis of metagenomic data

The source code for this pipeline is available from https://github.com/mhnb/camamed.

Software Requirements

- Linux operating system (Preferably Ubuntu)
- Python 2 or 3 (Preferably Python ≥3.7)
- All required software will be installed automatically after executing the *python camamed_init.py* command, otherwise, refer to Appendix A for manual installation.

Hardware Requirements

This software can run on a regular PC with 8GB RAM and a single processor. Of course, it depends on the size of the samples and the gene catalog, but for a run with the higher performance, it is better to run on a computer with at least 32GB of RAM and ten cores of processor.

Initialization steps before starting

- Copy the FASTA format gene catalog files in the /CAMAMED/ folder
- If the sequence files are SRA format
    - ➢ Copy SRA samples in the ~/sra_files folder.
    - ➢ Copy the SRA sample names in the ~/sra_files/sra_file_names.txt file.
        - ▪ for example (sra_file_names):

          file1.sra
          file2.sra
          file3.sra
    - ➢ After executing SRA-Toolkit, Fastq or Fasta files are automatically copied to folder ~/Read_files.
- Copy Fastq or Fasta samples in the ~/Read_files folder (If the format of the input sequences is not SRA).
- Copy the sample names in the ~/Read_files/sample_file_names.txt file and label them in the ~/Read_files/class_label.txt file.
    - ➢ for example (sample_file_names for paired-end sequences):

      p_file1_1.fq
      p_file1_2.fq
      p_file2_1.fq
      p_file2_2.fq
      p_file3_1.fq
      p_file3_2.fq

  - ➢ for example (class_label for paired-end sequences):
    class1
    class2
    class3
  - ➢ If the samples are paired-end, enter a label for two files that are typed in sequence.
- A sample of the gene catalog is located in the following folder:
  - ➢ ~/sample_input_files/gene_catalog
- Some samples, with their names and labels, could be found in the following folder:
  - ➢ ~/sample_input_files/Read_files
- For example, KAAS and GhostKOALA outputs are in the following folder:
  - ➢ ~/sample_input_files/gene_ko_outputs

Also, the results of executing commands with default parameters on the ~/sample_input_files folder data are in the ~/sample_output_files folder.

Order of the execution of functions for using the CAMAMED pipeline

Figure 1 shows the sequence of functions for analyzing metagenomic data using the CAMAMED pipeline in two different ways.

Important points

Point1: Note that if you have a storage limitation to store the files of sequences, you can copy permitted number of files in the ~/sra_files or ~/Read_files folder and save the names of the copied files in ~/sra_files/sra_file_names.txt or ~/Read_files/sample_file_names.txt files and run below functions:

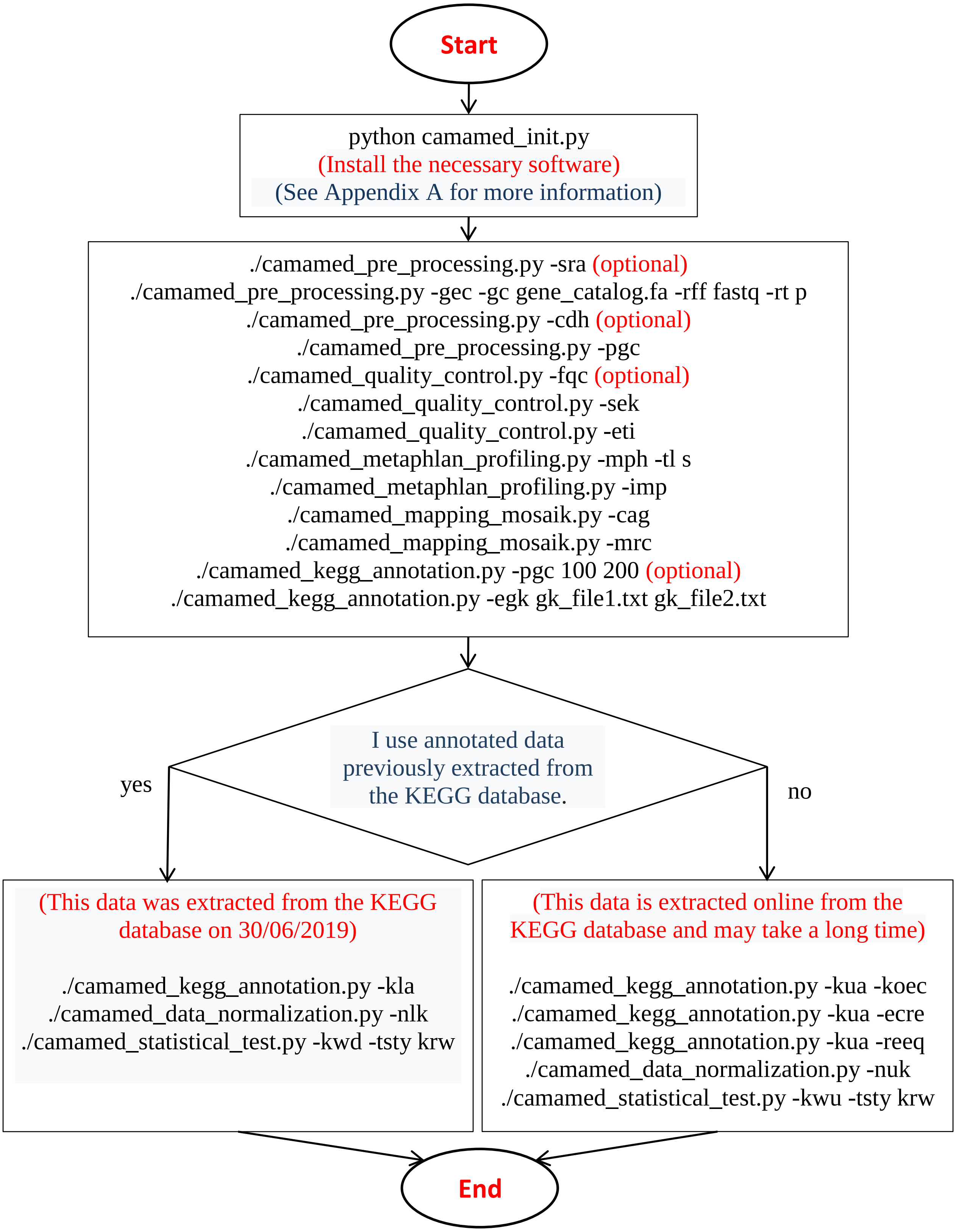
- ./camamed_pre_processing.py -sra (optional)
- ./camamed_quality_control.py -fqc (optional)
- ./camamed_quality_control.py -sek
- ./camamed_metaphlan_profiling.py -mph -tl s
- ./camamed_mapping_mosaik.py –cag

Continue this step until all samples are completed.

Point2: The below functions are performed on the gene catalog, and in any case, should be done at the beginning of the work.

- ./camamed_pre_processing.py -gec -gc gene_catalog.fa -rff fastq -rt p
- ./camamed_pre_processing.py -cdh (optional)
- ./camamed_pre_processing.py -pgc
- ./camamed_mapping_mosaik.py -cag

Point3: After running Point1 on all samples and Point2, enter the sample's name in the ~/Read_files/sample_file_names.txt file as described in the 'Initialization steps before starting' section and run the remaining functions that are related to the whole sequences.



```
python camamed_init.py
```

(Install the necessary software)

(See Appendix A for more information)

```
./camamed_pre_processing.py -sra (optional)
./camamed_pre_processing.py -gec -gc gene_catalog.fa -rff fastq -rt p
./camamed_pre_processing.py -cdh (optional)
./camamed_pre_processing.py -pgc
./camamed_quality_control.py -fqc (optional)
./camamed_quality_control.py -sek
./camamed_quality_control.py -eti
./camamed_metaphlan_profiling.py -mph -tl s
./camamed_metaphlan_profiling.py -imp
./camamed_mapping_mosaik.py -cag
./camamed_mapping_mosaik.py -mrc
./camamed_kegg_annotation.py -pgc 100 200 (optional)
./camamed_kegg_annotation.py -egk gk_file1.txt gk_file2.txt
```

I use annotated data previously extracted from the KEGG database.

(This data was extracted from the KEGG database on 30/06/2019)

```
./camamed_kegg_annotation.py -kla
./camamed_data_normalization.py -nlk
./camamed_statistical_test.py -kwd -tsty krw
```

(This data is extracted online from the KEGG database and may take a long time)

```
./camamed_kegg_annotation.py -kua -koec
./camamed_kegg_annotation.py -kua -ecre
./camamed_kegg_annotation.py -kua -reeq
./camamed_data_normalization.py -nuk
./camamed_statistical_test.py -kwu -tsty krw
```

Figure 1: shows the sequence of functions for analyzing metagenomic data using the CAMAMED pipeline.

CAMAMED functions

1- *python camamed_init*

This function performs the initial setting and installation of related applications to run CAMAMED. This function is executed with the system's default python that is better to have Python version 3.7 or higher. But it runs with other versions and even python 2.7. If the required applications are not automatically installed, Appendix A can be used for manual installation.

For example: python camamed_init

2- *camamed_pre_processing*

This function is a preprocessing step in pipeline

Command: ./camamed_pre_processing.py method [options]

Methods:

-h Shows help related to this function

-sra Running SAR toolkit to convert SRA files to Fastq or Fasta files. (**optional**)

Copy SRA samples in the ~/sra_files folder.

Copy the sample names in the ~/sra_files/sra_file_names.txt.

For example:

```
file1.sra
file2.sra
file3.sra
```

After executing SRA-Toolkit, Fastq or Fasta files are automatically copied to folder ~/Read_files.

For Example: ./camamed_pre_processing.py -sra

-gec Get information on sequences and gene catalogs.

Options:

-gc (gene_catalog): Gene catalog sequences must have Fasta format.

First copy the gene catalog into the /CAMAMED folder.

-rff (read_files_format)[fastq/fasta]:

Copy the sample files in the ~/Read_files folder.

Enter the file name of the samples in the ~/Read_files/sample_file_names.txt

For single end Fastq files:

```
file1.fastq
file2.fastq
file3.fastq
```

For paired end Fastq files:

```
file1_1.fastq
file1_2.fastq
file2_1.fastq
file2_2.fastq
```

-rt (read_type): paired end or single end [p/s]

-is (insert_size): For paired end sequences (An integer number) or ignor enter -1 (Default=-1).

For Example:

```
./camamed_pre_processing.py -gec -gc gene_catalog.fa -rff fastq -rt p
./camamed_pre_processing.py -gec -gc gene_catalog.fa -rff fastq -rt p -is 350
```

-cdh Running CD-HIT on the gene catalog to remove redundant genes. (**optional**)

You can use this option if you think your gene catalog has redundant sequences.

After deleting redundant genes, the new gene catalog is saved with the name cd_hit_gene_catalog. Also, clustered genes are saved in a file named cd_hit_gene_catalog.clstr, and the genes of the head cluster are marked with asterisk (*).

Options:

-sit (sequence_identity_threshold): This value can be in the range of 0.8 to 1 (Default=0.9).

For Example: ./camamed_pre_processing.py -cdh

./camamed_pre_processing.py -cdh -sit 0.95

-pgc Peprocessing gene catalog.

At this point, the names of the genes are deleted from the gene catalog and stored in the ~/files/gene_name.txt and for the genes the gene1, gene2 and ... are respectively selected.

For Example: ./camamed_pre_processing.py –pgc

3- ***camamed_quality_control***

This function executes Fastqc quality control and SeqKit tools on sample sequences.

At this state, if the sequences have Fastq format, they will be quality controlled and their statistical information extracted. But if they have Fasta format, only their statistical information obtained. Before running this step, the sample data should be in the ~/Read_files folder and the file names should be written in the text file ~/Read_files/sample_file_names.txt, respectively.

Command: ./camamed_quality_control.py method

Methods:

-h Shows help related to this function.

-fqc Execut FastQC quality control only for Fastq files (**optional**).

This method is used to control the quality of Fastq sequences. In this step, FastQC software executes on sequences and the outputs saved as HTML files in the ~/fastqc_output/ folder.

For Example: ./camamed_quality_control.py -fqc

-sek Execut SeqKit to extract information from sample files.

At this point, SeqKit software is run to extract the statistical information of the samples, and results are saved in ~/seqkit_output folder.

For Example: ./camamed_quality_control.py -sek

-eti Extract total information from SeqKit outputs.

In this step, all the statistical information related to the Seqkit outputs is extracted and saved in the ~/all_results/total_sample_info.txt file.

For Example: ./camamed_quality_control.py –eti

4- *camamed_metaphlan_profiling*

In this function, using the Metaphlan2 software extract the abundance of all bacteria. Metaphlan2 can produce taxonomic profiling at different levels. Such as Kingdom, Phylum, Class, Order, Family, Genus, and Species.

Command: ./camamed_metaphlan_profiling.py method [options]

Methods:

-h Shows help related to this function.

-mph Execute metaphlan2.

By choosing this option, the MetaPhlan2 software runs on samples, and the results are stored in the ~/metaphlan_output folder. You can use the 'metaphlan2' command to access MetaPhlan2 help. Meanwhile, MetaPhlan2 only returns the results of prokaryotic genomes and ignores the genomic information of eukaryotes, viruses, and archaea. For more configurations, refer to the ~/metaphlan_samlpe.sh file.

Options:

-tl (taxa_level): For selecting Kingdom, Phylum, Class, Order, Family, Genus, or Species, Select one of {'k', 'p', 'c', 'o', 'f', 'g', 's'}

-c (core_number): select number of cores for Metaphlan2 execution (Default=1). The number of cores must be a positive integer, otherwise one is chosen.

For Example: ./camamed_metaphlan_profiling.py -mph -tl s -c 3

-imp Extract information from metaphlan2 output files.

At this step, information about the bacteria is selected at the taxonomic level, and their frequency is calculated (Frequency is reported as percentages) and stored in the ~/all_results/total_metaphlan_results.txt file. If the sequences are paired-end, instead of the two files per one sample, only one output is reported as an average.

For Example: ./camamed_metaphlan_profiling.py –imp

5- *camamed_mapping_mosaik*

This function maps the reads to the genes catalog using the MOSAIK software. To mapping the read sequences to the gene catalog, the following two steps should be implemented. In the first step, the gene catalog should be converted into an acceptable form for the MOSAIK software. In the second step, the reads are mapped to the gene catalog.

Command: ./camamed_mapping_mosaik.py method [options]

Methods:

-h Shows help related to this function

-cag Creating an acceptable form of gene catalog.

At this stage, two MosaikBuild and MosaikJump tools are run on the gene catalog to prepare it for sequence mapping. To access the help of these tools, you can run './MosaikBuild -h', and './MosaikJump -h' commands in the Linux terminal at

/CAMAMED/ path, and you can refer to the ~/mosaik_build_ref.sh file for further configuration.

Options:

-hw (hash word): Select the length of the hash word that can be in range 4 to 32 (Default=15).

```
For Example: ./camamed_mapping_mosaik.py -cag
             ./camamed_mapping_mosaik.py -cag -hw 17
```

-mrc Mapping reads to the gene catalog using MOSAIK software.

At this state, the MosaikBuild tool is used for preparing the sequences and the MosaikAligner tool is used to map the sequences to the gene catalog prepared in the previous step. To access the help of these tools, the './MosaikBuild -h' and './MosaikAligner -h' commands are executed at /CAMAMED/ path. Refer to the ~/mosaik_read_aligner.sh file for more configurations. The mapping results are stored in a SAM format in the ~/mosaik_outputs folder.

Options:

-c (core_number): Select number of cores for MOSAIK execution (Default=1).
The number of cores must be a positive integer.

```
For Example: ./camamed_mapping_mosaik.py -mrc
             ./camamed_mapping_mosaik.py -mrc -c 5
```

6- ***camamed_kegg_annotation***

This function extracts annotated information from KEGG databases.

Command: ./camamed_kegg_annotation.py method [options]

Methods:

-h Shows help related to this function

-pgc Preparing the Gene Catalog for extracting annotated information from the KEGG databases. To obtain KOs associated with gene sequences using web services GhostKOALA and KAAS, it would be better if the file size of the gene catalog is less than 300 MB. The gene catalog is stored after the preprocessing and deletion of the duplicated genes (optional) named main_gene_catalog.fa in the /CAMAMED/ folder. If the gene catalog size is more than 300MB, you can use this option to convert it to smaller files. For example, if the gene catalog has 300 gene sequences, you can convert it to three files with 100 genes by entering the values 100 200. This will split the gene catalog into three files, each with 100 genes. If the file size does not get smaller than 300MB, this step should be re-run. Finally, the smaller files are located in the ~/sub_catalog_files folder and are ready to be uploaded to the web service.

After the conversion of the gene catalog into the files smaller than 300MB, for nucleotide sequences, both the KAAS and the GhostKOALA web services can be used to obtain KOs associated with each gene sequence. However, for amino acids, only GhostKOALA can be used to get KOs.

- Link to the KAAS web service for uploading sequences

- ➢ https://www.genome.jp/kaas-bin/kaas_main

- Link to the GhostKOALA web service for uploading sequences

- ➢ https://www.kegg.jp/ghostkoala/

For Example: ./camamed_kegg_annotation.py -pgc 100 200

-egk Extracting information from GhostKOALA or KAAS output files.

In this step, the files generated by GhostKOALA or KAAS server are read and the required information is extracted. Two examples of the KASS and the GhostKOALA web services output are in the ~/sample_input_files/gene_ko_outputs folder. If the gene catalog is more than one file, the results should also be saved in a few text files, and the order in which they will be uploaded to the CAMAMED will be the same as the order of gene number. To continue running, Web services output files must be copied to the ~/kegg_annotation folder. At this point, the names of the files should be entered, for example, ' gk_file1.txt gk_file2.txt'. The ordering of the files is based on the gene number.

After running this function, two ko.txt and gene_ko.txt files are created in the ~/kegg_annotation folder, in which the entire KOs in the samples and the relationship between the genes and the KOs are determined, respectively.

For Example: ./camamed_kegg_annotation.py -egk gk_file1.txt gk_file2.txt

-kla KEGG local annotation

We extracted all the annotated information related to the KOs and EC numbers and reactions to the date 2019/6/30 from the KEGG database and saved in the /kegg_annotation/ folder in the text files that start with the 'def' prefix.

- EC numbers related to KOs and all EC numbers in the KEGG database are stored in def_ko_ec.txt and def_ec.txt files, respectively.
- Reactions related to EC numbers and all reaction numbers in the KEGG database are stored in def_ec_re.txt and def_re.txt files, respectively.
- Finally, the reaction definitions and the equation for each reaction are stored in the def_re_eq.txt file

After this step, two other files are created. The EC numbers associated with each gene and the reactions associated with each gene are stored in gene_ec.txt and gene_re.txt files, respectively.

For Example: ./camamed_kegg_annotation.py -kla

-kua KEGG user annotation

If you do not want to use the previously extracted data from the KEGG database, you can use this method. There are three options at this stage to be executed in sequence. But if you want to extract this information yourself, you can use these options.

****These steps may take a long time****

Options:

-koec Extract KO-related EC numbers from the KEGG database. At this step, all EC numbers associated with the ko.txt file are extracted online from the KEGG database and stored in separate files called ec.txt and ko_ec.txt. Also, the relationship between genes and EC numbers is stored in the gene_ec.txt file.

-ecre Extract EC-related reactions from the KEGG database. At this step, all reactions associated with the ec.txt file are extracted online from the KEGG database and stored in separate files called re.txt and ec_re.txt. Also, the relationship between genes and reactions is stored in the gene_re.txt file.

-reeq Extract reaction-related equation from the KEGG database. At this state, the definitions and equations for the reaction of the re.txt file are extracted online from the KEGG database and stored in the re_eq.txt file.

For Example: ./camamed_kegg_annotation.py -kua -koec
./camamed_kegg_annotation.py -kua -ecre
./camamed_kegg_annotation.py -kua -reeq

7- *camamed_data_normalization*

This function Extract samples information in normalized bacteria, gene, KO, EC number and reaction matrix.

Command: ./camamed_data_normalization.py method [options]

Methods:

-h Shows help related to this function

-nlk Normalizing data using local information previously extracted from the KEGG database. At this stage, the abundance of bacteria and genes, as well as KOs, EC numbers and reactions that are related to the identified genes in the previous sections, are normalized based on the CSS algorithm presented in the main paper and stored in the ~/all_results folder. The names of these files are normal_matrix_metaphlan.txt, normal_matrix_gene.txt, normal_matrix_ko.txt, normal_matrix_ec.txt and normal_matrix_re.txt respectively.

Options:

-mtct Minimum total counts for each taxon. This value is the percentage of all taxa, for example, 0.5 (Default=0.001).

-mtcg Minimum total counts of mapped reads per gene in total samples (Default=5).

For Example: ./camamed_data_normalization.py -nlk
./camamed_data_normalization.py -nlk -mtct 0.002 -mtcg 10

-nuk Normalizing data using information extracted by the user from the KEGG database. This step is exactly like '-nlk' method, the normalized data is generated based on the online information extracted by the user and saved with the same name in the ~/all_results folder.

Option:

-mtct Minimum total counts for each taxon. This value is the percentage of all taxa, for example, 0.5 (Default=0.001).

-mtcg Minimum total counts of mapped reads per gene in total samples (Default=5).

For Example: ./camamed_data_normalization.py -nuk
./camamed_data_normalization.py -nuk -mtct 0.002 -mtcg 10

8- *camamed_statistical_test*

This function performs the statistical test (Kruskal-Wallis H-test or ANOVA test) on normalized data. If the distribution of data is normal, the ANOVA test can be used, otherwise, the Kruskal-Wallis H-test can be used. At this stage, we perform the statistical test on normalized bacteria, gene, KO, EC number, and Reaction data that stored in the ~/all_results folder. To get started, you first select the label of class for each sample in the ~/Read_files/class_label.txt file. The number of classes can be in the range [2:10]. For each sample, enter a separate row. For example:

```
class1
class2
class1
class3
```

Command: ./camamed_kruskal_wallis_test.py method [options]

Methods:

```
-h    Shows help related to this function
-kwd  Running the statistical test on the default annotated data.
      Option:
      -tsty  Statistical type (Kruskal-Wallis H-test or ANOVA test)[krw/ano]
      -pval  The p-value to filter the output. This value can be in the interval [0:1] (Default=0.05).
      For Example: ./camamed_statistical_test.py -kwd -tsty krw
                   ./camamed_statistical_test.py -kwd -tsty ano
                   ./camamed_statistical_test.py -kwd -tsty krw -pval 0.01
                   ./camamed_statistical_test.py -kwd -tsty ano -pval 0.01
      -kwu  Running the statistical test on the user extracted data.
      Option:
      -tsty  Statistical type (Kruskal-Wallis H-test or ANOVA test)[krw/ano]
      -pval  The p-value to filter the output. This value can be in the interval [0:1] (Default=0.05)
      For Example: ./camamed_statistical_test.py -kwu -tsty krw
                   ./camamed_statistical_test.py -kwu -tsty ano
                   ./camamed_statistical_test.py -kwu -tsty krw -pval 0.01
                   ./camamed_statistical_test.py -kwu -tsty ano -pval 0.01
```

Appendix A

Manual installation in python 3

- MetaPhlAn2
  - ➢ Installation command ('sudo apt install metaphlan2')
  - ➢ After the first run, MetaPhlAn database files are downloaded automatically. Otherwise, download the files from the following links and copy them to the installation path in folder /usr/share/metaphlan2/databases.
    - ▪ https://bitbucket.org/biobakery/metaphlan2/downloads/mpa_v20_m200.tar
    - ▪ https://bitbucket.org/biobakery/metaphlan2/downloads/mpa_v20_m200.md5
- CD-HIT
  - ➢ Installation command ('sudo apt-get install cd-hit')
- SRA-Toolkit
  - ➢ Installation command ('sudo apt install sra-toolkit')
- Samtools
  - ➢ Installation command ('sudo apt-get install samtools')
- FastQC
  - ➢ Installation command ('sudo apt install fastqc')
- Also install the necessary packages for Python if requested. For example:
  - ➢ sudo apt install python3-pip
  - ➢ sudo pip3 install pandas
  - ➢ sudo pip3 install numpy
  - ➢ sudo pip3 install scipy
  - ➢ sudo pip3 install biopython
- If you also use Python 2, use the following commands to install the above packages.
  - ➢ sudo apt install python-pip
  - ➢ sudo pip install pandas
  - ➢ sudo pip install numpy
  - ➢ sudo pip install scipy
  - ➢ sudo pip install biopython