DESeq2_pipeline

Release 1.0

Gabor Balogh

May 19, 2021

CONTENTS:

"DESeq2_pipeline" a tool to choose, harvest and analyse expression data of the TCGA-projects with help of the DESeq2 R package

build and activate the provided conda env:

```
$ conda env create -f deseq_env.yaml
$ conda activate deseq_pipeline
```

call the script without any options to enter the interctive mode and set each option step by step:

```
$ python main_deseq.py
```

print help page:

```
$ python main_deseq.py --help
```

Usage: main_deseq.py [OPTIONS]

Options:

-D, --download_data perform download and merging steps, without the DESeq2 analysis

-A, --analyse_data perform the DESeq2 analysis

-o, --out_path TEXT The Path, where the results are saved to [default: CWD]

-s, --script_path TEXT The Path, where the DESeq2_pipeline script is located, if not supplied, the current working directory is assumed

-d, --drugs TEXT drug(s), like: -d drug1 -d drug2 or drugcombination(s), like: -d drug1,drug2

-p, --project TEXT Project, that shall be applied, choose between TCGA- CESC, TCGA-HNSC, TCGA-LUSC, TCGA-ESCA, or combinations out of them like -p project1 -p project2

-f, --function INTEGER running functions seperately lookup the manpage for detailed description of each function

-t, --treshold INTEGER treshold for a higher division of the kaplan meier plots [default: 0]

--help Show this message and exit.

main_deseq.**call_with_options**(**args*, ***kwargs*)

"DESeq2_pipeline" a tool to choose, harvest and analyse expression data of the TCGA-projects with help of the DESeq2 R package

build and activate the provided conda env:

$ conda env create -f deseq_env.yaml

$ conda activate deseq_pipeline

call the script without any options to enter the interctive mode and set each option step by step:

$ python main_deseq.py

print help page:

$ python main_deseq.py –help

choose_therapy.**Choose_drugs**(*SCRIP_PATH*, *PROJECTS*)

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param PROJECTS: list of projects choosen

Type PROJECTS: list of str

interactively requesting the drugs which shall be applied to the deseq approach

choose_therapy.**Choose_path_and_option**(*OUTPUT_PATH*, *PROJECTS*, *DRUGS*, *function*, *SCRIPT_PATH*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param PROJECTS: list of projects choosen

Type PROJECTS: list of str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param function: applied functions

Type function: int

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

interactively requesting whether download steps, analysis or both should be performed

choose_therapy.**Choose_project**()

interactively requesting the Projects that shall be applied to the deseq approach

create_matrix_new.**correct_drugs**(*PROJECT*, *OUTPUT_PATH*, *logger*)

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

creating table: 'OUTPUT_PATH/DF_3t_both_with_DRUG_combi.tsv', drugcombination in field 'drugnames' as ordered set, comma seperated. header includes: UUID case_id gender vital_status drugnames survivaltime years_to_last_follow_up year_of_birth year_of_death age_at_diagnosis PROJECT

```
# for executing this step via terminal, issue -f 6 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 6
```

create_matrix_new.**create_snake_config**(*OUTPUT_PATH*, *PROJECT_title*, *DRUGS_title*, *project_list*, *DRUGS*, *SCRIPT_PATH*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param PROJECT_title: concatenated str of multiple projects

Type PROJECT_title: str

Param DRUGS_title: concatenated str of multiple drugs

Type DRUGS_title: str

Param project_list: list of applied projects

Type project_list: list of str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

out of the log files in PROJECT_title/DRUGS_title/test_log.log parse out all outputfiles of the applied run and create PROJECT_title/DRUGS_title/snakemake_config.yaml

```
# for executing this step via terminal, issue -f 15 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 15
```

create_matrix_new.**create_statistics_from_DESeq2_tables**(*OUTPUT_PATH*, *DRUGS*, *SCRIPT_PATH*, *PROJECT*, *logger*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

adding some statistics to the result output tables from DESeq2 table is saved in OUTPUT_PATH/PROJECT_title/DRUGS_title/DESeq2_out***/results_statistics.tsv

```
# for executing this step via terminal, issue -f 8 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 8

# when choosing multiple projects, call:
$ python main_deseq.py -p TCGA-CESC  -p TCGA-HNSC -d cisplatin -f 10
```

create_matrix_new.**create_summary_table**(*OUTPUT_PATH*, *PROJECT*, *logger*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

get the raw data out of the gzip compressed files and merge them together, save complete merged table in OUTPUT_PATH/PROJECT/summary_DF.tsv

Table 1: example structure for summary_DF.tsv:

gene identifier	case_id with counts
genes	6ff12a54-10da-4941-bfea-7b66e19b4be9 ...
ENSG00000000003	3423 ...
ENSG00000000005	0 ...

```
# for executing this step via terminal, issue -f 5 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 5
```

create_matrix_new.**download_clinical_tables**(*UUID*, *PROJECT*, *OUTPUT_PATH*, *logger*)

Param UUID: unique file identifier of the meta table

Type UUID: str

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

with the UUID the clinical tables will be downloaded in the OUTPUT_PATH/PROJECT:

- nationwidechildrens.org_clinical_patient_****.txt
- nationwidechildrens.org_clinical_drug_****.txt

```
# for executing this step via terminal, issue -f 2 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 2
```

create_matrix_new.**download_data_files**(*PROJECT*, *FILE_TYPE*, *OUTPUT_PATH*, *logger*)

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param FILE_TYPE: type of raw data to download from TCGA

Type FILE_TYPE: str

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

dowloading all data files and the belonging manifest with UUID to each file creating a subdir in the OUTPUT_PATH/PROJECT/ folder named "{PROJECT}_data_files" the belonging MANIFEST.txt is saved in OUTPUT_PATH/PROJECT

```
# for executing this step via terminal, issue -f 3 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 3
```

create_matrix_new.**merging_meta_infos**(*OUTPUT_PATH*, *PROJECT*, *logger*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

merging the infos out of the 3 tables together(nationewidechildrens.org.., manifest, metainfo) creating 3 new tables, one with innerjoin(both), two with left and right outer join to save those files where information is missing: * DF_3t_left: no information about therapeutic_agents * DF_3t_right: missing filename (case_id present)

- DF_3t_both holds infos like UUID, filename, md5, size, state, case_id, gender, vital_status, year_of_birth, year_of_death and some more..

every table saved in OUTPUT_PATH/PROJECT/

```
# for executing this step via terminal, issue -f 4 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 4
```

create_matrix_new.**meta_filter**(*PROJECT*, *FILE_TYPE*, *OUTPUT_PATH*, *logger*)

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param FILE_TYPE: type of raw data to download from TCGA

Type FILE_TYPE: str

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

creating the meta_info.dat file in your OUTPUT_PATH/PROJECT directory

```
# for executing this step via terminal, issue -f 1 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 1
```

create_matrix_new.**provide_DESeq2_table**(*PROJECT*, *OUTPUT_PATH*, *DRUGS*, *SCRIPT_PATH*, *logger*)

Param PROJECT: list of projects choosen

Type PROJECT: list of str

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param logger: the adjustet logger with the right filehandler

Type logger: logging instance

filtering case_id according to DRUG query providing new table for DESeq2 analysis dependend on distinguishable factors of the tables provided, a singlefactorial (at least differences in vital state) or a mutlifactorial run in DESeq2 is performed (gender, therapy or project)

```
# for executing this step via terminal, issue -f 7 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 7

# when choosing multiple projects, call:
$ python main_deseq.py -p TCGA-CESC  -p TCGA-HNSC -d cisplatin -f 9
```

create_matrix_new.**set_logger**(*OUTPUT_PATH*, *PROJECT_title*, *DRUGS_title*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param PROJECT_title: merged project title out of multiple projects

Type PROJECT_title: str

Param DRUGS_title: merged drug title out of multiple drugs

Type DRUGS_title: str

every paths and options are set, configure here the logfiles, with which the snakemake config files are going to be created we create 2 loggers, in case just one project is applied the logs are written in PROJECT/DRUGS_title/test_log.log in case multi project is applied, the logs are written in PROJECT_title/DRUGS_title/test_log.log with that, it is clear which config file shall be createt out of the logfiles present in one outputpath (the drugs path must therefore be created from the first fct, to write the log file also, the dir of the logfile must be logged, s.t. snakemake knows where the input file for the final snakemake configuratioin file is located

class lifeline_summary_test_2.**Lifeline_plot**(*OUTPUT_PATH*, *PROJECT_title*, *DRUGS_title*, *path_to_result*, *prefix*, *treshold*, *multi_project*)

returning the ENSG on which the gene specific KaplanMeier plots are created

lifeline_summary_test_2.**lifelines_ENSG**(*OUTPUT_PATH*, *DRUGS*, *PROJECT_DRUG_UUID*, *treshold*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of strings

Param PROJECT_DRUG_UUID: hash holding a project to the UUID of the belonging drugtable

Type PROJECT_DRUG_UUID: dict

Param treshold: parameter for the lifeline plots helping for the classification of expression data

Type treshold: int

script creates the plots:
lifelines_cumulative_density.svg,
lifelines_multiple_groups.svg,
lifelines_parametric_models_2.svg,
lifelines_survival_fct.svg,
lifelines_table.tsv in OUTPUT_PATH

- with the different expression ENSGs (normalized counts given from deseq, median from them and separated in UP and DOWN) groups plottet with lifeline

needed:

- ALL_PROJECTS_summary_dead_alive_reduced_INFO.tsv as

count_DF_MI (just the info for building the multiindex for **count_DF**):

Table 2: example multiindex table

variable	value
vital_status	alive
gender	female
case_id	6ff12a54-10da-4941-bfea-7b66e19b4be9
PROJECT	TCGA-CESC

- DESeq2_MF_normalized_counts_reduced.tsv as

Table 3: ENSG table, **count_DF** (normalized counts with help of DESeq2)

variable	value
ENSG00000000003	4109.0073147311
ENSG00000000005	0

- OUTPUT_PATH/DRUGS/DESeq2_MF_results_reduced.tsv as **DF_res**

creates the **ENSG_list** on the basis of the DESeq_results in dependence of the resulttables, we get the 10 highest and 10 lowest logfoldchange

–> here we sort log2fold change wise and merge then the most different INCREASING and DECREASING cases with the TCGA-*/DF_3t_both_with_DRUG_combi.tsv, in this table we have the fields:

UUID case_id gender, vital_status, drugnames, survivaltime, years_to_last_follow_up, year_of_birth, year_of_death, age_at_diagnosis, PROJECT,

with it we create the table for lifeline

Table 4: example table for lifeline input

index	T	E	case_id
0	5.197260273972604	True	9ffa79fa-d2d8-48e1-8fd6-4b020ecf357c
1	0.8602739726027397	False	0de19185-3517-4e30-925b-7eb1f5079ec2

- the up and down separation depends on the median of the normalized count matrix
- setting the threshold based on the median of the logfoldchange, delete out 10 % around it if -f 10 is applied

```
# for executing this step via terminal, issue -f 11 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 11

# you can try out different tresholds, new directorys are
# created therefore:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 11 -t 10
```

walk_all_drug_frequency.**drug_frequency**(*OUTPUT_PATH*, *DRUGS*, *SCRIPT_PATH*, *PROJECT_DRUG_UUID*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param PROJECT_DRUG_UUID: hash holding a project to the UUID of the belonging drugtable

Type PROJECT_DRUG_UUID: dict

create an overview of all available drugs, of the applied projects in DESeq2 output dir

```
# for executing this step via terminal, issue -f 12 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 12
```

walk_all_drug_frequency.**drug_frequency_all_single_projects**(*OUTPUT_PATH*, *DRUGS*, *SCRIPT_PATH*, *PROJECT_DRUG_UUID*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param PROJECT_DRUG_UUID: hash holding a project to the UUID of the belonging drugtable

Type PROJECT_DRUG_UUID: dict

take every drug frequency out of the single project folders, therefore walk in the OUTPUT_PATH/TCGA-[2..4] folders. . .

```
# for executing this step via terminal, issue -f 14 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 14
```

create_report.**create_report_pdf**(*OUTPUT_PATH*, *DRUGS*, *SCRIPT_PATH*, *PROJECT_DRUG_UUID*, *treshold*)

Param OUTPUT_PATH: path for DESeq2 pipeline outputs

Type OUTPUT_PATH: str

Param DRUGS: applied drug(s)

Type DRUGS: list of str

Param SCRIPT_PATH: path to the DESeq2_pipeline repo

Type SCRIPT_PATH: str

Param PROJECT_DRUG_UUID: hash holding a project to the UUID of the belonging drugtable

Type PROJECT_DRUG_UUID: dict

Param treshold: parameter for the lifeline plots helping for the classification of expression data

Type treshold: int

creating a report file with all visual outputs created in an analysis, saved at OUTPUT_PATH/PROJECT_title/DRUGS_title/REPORT.pdf

```
# for executing this step via terminal, issue -f 13 to your call,
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 13 -t 0
# if you want to include several tresholds you had created in prior
# with the KaplanMeier function (-f 11 or within -A), apply them here
# as well, for
# example:
$ python main_deseq.py -p TCGA-CESC -d cisplatin -f 13 -t 0 -t 50 -t
100
```

CHAPTER

ONE

TABLES

- *example structure for summary_DF.tsv:*
- *example multiindex table*
- *ENSG table, count_DF (normalized counts with help of DESeq2)*
- *example table for lifeline input*

CHAPTER

TWO

DIRECT FUNCTION APPLOYMENT:

- function_1
- function_2

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX