Developer Level Documentation

Bash

Bash scripts that acts like a wrapper around the different functions of this workflow.

`analysis_simulation.sh`

Bash script for performing a simulation study.

Usage

```
bash analysis_simulation.sh -t <treename>.nwk -m JC -n 1000 -k 100  -s true
```

Parameters

- `-t` - tree file in Newick format `<treename>.nwk`
- `-m` - substitution model for sequence simulation `<m>`, by default JC
- `-n` - sequece length `<n>`, by default 500
- `-k` - number of simulations `<k>`, by default 1
- `-s` - apply staturation tests true or false `<s>`, by default false

Structure

1. read in the command line arguments
2. create directory `results_<treename>`, where all results will be saved
3. create a new directory `alignment_alisim`
4. call the IQ-Tree command (save command line output to `simulation.log`):

```
  iqtree2 –alisim alignment -m <model> -t <treename>.nwk –length <n> –num-
alignments <k>
```

5. loop trough all alignment simulation files in `alignment_alisim`
6. apply C++ script by using command:

```
  ./all_tests.out -F <treename>-alignment_*.phy -s <s>
```

7. save raw test results in `results_<treename>/results_raw_<treename>.csv`
8. call R script on the raw test results:

```
  analysis_visualisation_simulation.R <treename>.nwk <n>
results_<treename>/results_raw_<treename>.csv <k> <s>
```

9. delete directory with simulated alignments `alignment_alisim`

`analysis_biological.sh`

Bash script for performing a biological study.

Usage

```
bash analysis_biological.sh -a <alignment>.phy -t <treename>.nwk -s true
```

or

```
bash analysis_biological.sh -a <alignment>.phy -iqtr -m GTR -s true
```

Parameters

- `-a` - multiple sequence alignment file in Phylip or NEX format `<alignment>.phy`
- `-t` - tree file in Newick format `<treename>.nwk`
- `-iqtr` - specifies whether to call IQ-TREE to compute the ML tree
- `-m` - substitution model for IQ-TREE to use if `-iqtr`flag present `<m>`, by default none (if none IQ-TREE ModelFinder will be called)
- `-s` - apply staturation tests true or false `<s>`, by default false

Structure

1. read in the command line arguments
2. create directory `results_<alignment>`, where all results will be saved
3. copy the alignment file in this directory
4. call the C++ script by using command:

```
./all_tests.out -F <alignment>.phy -s <s>
```

5. save raw test results in `results_<alignment>/results_raw_<alignment>.csv`
   6.1. if no tree file or no `-iqtr`flag provided - end here
   6.2. if option `-iqtr` provided, create a new directory `<alignment>_IQTREE`, move .phy there and call IQ-TREE by using command:

```
iqtree2 -s <alignment>.phy -m <m> --redo-tree -quiet
```

7. call R script on the raw test results using the provided tree file or the one inferred with IQ-TREE:

```
  analysis_visualisation_biological.R <treename>.nwk
results_<treename>/results_raw_<treename>.csv <s>
```

C++

Sequence

Class for storing sequences.
Location: `bin\lib\Sequence.h`

Attributes

- `[std::string]` **id** - the name of the sequence
- `[std::string]` **seq** - the nucleotide sequence
- `[double]` **length** - length of the sequence
- `[Eigen::Vector4d]` **nucl_freqs** - nucleotide frequencies of the sequence
- `[Eigen::Vector4d]` **nucl_freqs_align** - nucleotide frequencies of the sequence, counting only non-gaps in an alignment

Constructors

```
Sequence(std::string seq_id, std::string seq_str, Eigen::Vector4d seq_freq)
```

Initializes an object with a specified `id` - **seq_id**, nucleotide sequence `seq` - **seq_str** and a nucleotide frequency vector `nucl_freqs` - **seq_freq**

Alignment

Class for storing an alignment.
Location: `bin\lib\Sequence.h`

Attributes

- `[std::vector<Sequence>]` **sequences** - a vector that stores objects of class `Sequence`
- `[Eigen::Vector4d]` **global_freqs** - global nucleotide distribution for the whole alignment

Constructors

```
Alignment()
```

Default constructor, intializes an object with empty attributes.

seqs_read()

Function for reading in sequences from an alignment. It creates an `Alignment` object, storing all sequences as its attribute, as well as the global alignment. For each sequence it also calculates its nucleotide distribution.
Location: `bin\lib\file_handling.h`

```
seqs_read(std::string file_name, std::string extension)
```

Parameters

- `[std::string]` **file_name** - name of the input file
- `[std::string]` **extension** - file extension

Returns

An `Alignment` object.

div_mat()

Function for calculation the sample diversity matrix of two sequences.
Location: `bin\lib\div_mat.h`
Author: [Gubela, 2022]

```
div_mat(string seq1, string seq2)
```

Parameters

- `[std::string]` **seq1** - first sequence
- `[std::string]` **seq2** - second sequence

Returns

An `Eigen::Matrix4d` object.

bowker_stat()

Function for calculating the Bowker test as $m^T*B^{-1}*m$.
Location: `bin\lib\stats.h`
Author: [Gubela, 2022}

```
bowker_stat(Eigen::Matrix<double, 6, 1> m, Eigen::Matrix<double, 6, 6>  B)
```

Parameters

- `[Eigen::Matrix<double, 6, 1>]` **m** - vector with the absolute differences of the off diagonal entries of the sample diversity matrix
- `[Eigen::Matrix<double, 6, 6>]` **B** - diagonal matrix with the sums of the off diagonal entries of the sample diversity matrix

Returns

A `double` type.

get_m()

Function for calculating *m* for the computation of the Bowker statistic.
Location: `bin\lib\stats.h`
Author: [Gubela, 2022]

```
get_m(Eigen::Matrix4d H)
```

Parameters

- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences

Returns

An `Eigen::Matrix<double, 6, 1>` object.

get_B()

Function for calculating *B* for the computation of the Bowker statistic.
Location: `bin\lib\stats.h`
Author: [Gubela, 2022]

```
get_B(Eigen::Matrix4d H)
```

Parameters

- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences

Returns

An `Eigen::Matrix<double, 6, 6>` object.

stuart_stat()

Function for calculating the Stuart test as $D^T*V^{-1}*D$.
Location: `bin\lib\stats.h`
Author: [Gubela, 2022}

```
stuart_stat(Eigen::Matrix<double, 3, 1> D, Eigen::Matrix<double, 3, 3>  V)
```

Parameters

- `[Eigen::Matrix<double, 3, 1>]` **D** - vector calculated using the vector from **get_m()**
- `[Eigen::Matrix<double, 3, 3>]` **V** - matrix calculated using the matrix from **get_B()**

Returns

A `double` type.

get_V()

Function for calculating *V* for the computation of the Stuart statistic.
Location: `bin\lib\stats.h`
Author : [Gubela, 2022]

```
get_V(Eigen::Matrix<double, 6, 6> B)
```

Parameters

- `[Eigen::Matrix<double, 6, 6> ]` **B** - the matrix calculated using **get_B()**

Returns

An `Eigen::Matrix<double, 3, 3>` object.

get_D()

Function for calculating *D* for the computation of the Stuart statistic.
Location: `bin\lib\file_handling.h`
Author : [Gubela, 2022]

```
get_D(Eigen::Matrix<double, 6, 6> m)
```

Parameters

- `[Eigen::Matrix<double, 6, 1> ]` **m** - the vector calculated using **get_m()**

Returns

An `Eigen::Matrix<double, 3, 3>` object.

intsym_stat()

Function for calculating the Internal Symmetry test as *Bowker - Stuart*.
Location: `bin\lib\stats.h`

```
intsym_stat(double bowk, double stu)
```

Parameters

- `[double]` **bowk** - the Bowker test statistic
- `[double]` **stu** - the Stuart test statistic

Returns

A `double` type.

get_d()

Function for calculating differences of the products of the 4 different Kolmogorov cycles.
Location: `bin\lib\stats.h`
Author : [Gubela, 2022]

```
get_d(Eigen::Matrix4d H)
```

Parameters

- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences

Returns

An `Eigen::Matrix<double, 4, 1>` object.

get_var()

Function for calculating the variance of the $\mathbf{d_i}$ (differenes in the Kolmogorov cycles).
Location: `bin\lib\stats.h`
Author : [Gubela, 2022]

```
get_var(Eigen::Matrix<double, 4, 4> P, int N, int d1)
```

Parameters

- `[Eigen::Matrix4d]` **P** - frequency matrix, calculated as the sample diversity matrix of two sequences divided by the sequence length
- `[int]` **N** - the sequence length
- `[int]` **d1** - index of the $\mathbf{d_i}$ differences

Returns

A `double` type

get_covar()

Function for calculating the covariance of $\mathbf{d_i}$ and $\mathbf{d_j}$ (differences in the Kolmogorov cycles).
Location: `bin\lib\stats.h`
Author : [Gubela, 2022]

```
get_covar(Eigen::Matrix<double, 4, 4> P, int N, int d1, int d2)
```

Parameters

- `[Eigen::Matrix4d]` **P** - frequency matrix, calculated as the sample diversity matrix of two sequences divided by the sequence length
- `[int]` **N** - the sequence length
- `[int]` **d1** - index of the $\mathbf{d_i}$ difference
- `[int]` **d2** - index of the $\mathbf{d_j}$ difference

Returns

A `double` type

sat_test_cas1()

Function for calculating the Saturation Test by Cassius with global frequences
Location: `bin\lib\sat_tests.h`

```
sat_test_cas1(Eigen::Matrix4d H, int n, Eigen::Vector4d freqs)
```

Parameters

- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences
- `[int]` **n** - the sequence alignment length
- `[Eigen::Vector4d]` **freqs** - global nucletotide absolute frequencies

Returns

A `double` type.

sat_test_cas2()

Function for calculating the Saturation Test by Cassius with local frequences from the alignments
Location: `bin\lib\sat_tests.h`

```
sat_test_cas2(Sequence Seq1, Sequence Seq2, Eigen::Matrix4d H, int n)
```

Parameters

- `[Sequence]` **Seq1** - the first sequence
- `[Sequence]` **Seq2** - the second sequence
- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences
- `[int]` **n** - the sequence alignment length

Returns

A `double` type.

chi_test()

Function for calculating the Chi^2 Saturation Test by Cassius with global frequences
Location: `bin\lib\sat_tests.h`

```
chi_test(Sequence Seq1, Sequence Seq2, Eigen::Matrix4d H, int n)
```

Parameters

- `[Sequence]` **Seq1** - the first sequence
- `[Sequence]` **Seq2** - the second sequence
- `[Eigen::Matrix4d]` **H** - the sample diversity matrix of two sequences
- `[int]` **n** - the sequence alignment length

Returns

A `double` type.

R file: analysis_visualisation_simulation.R / analysis_visualisation_biological.R

Used for calculating p-values and decisions of test statistics and plotting the results with 4 different visualsations.

Usage

```
Rscript analysis.R <treefile> <n> <raw_test_statistcs> <k> <s>
```

Parameters

- `treefile` - treefile either in Newick format or
- `n`- sequence length
- `raw_test_statistcs` - path to .csv file with raw test statistics
- `k` - number of simulations
- `s` - true/false saturation test

Includes functions:

heat_success()

Function for plotting a heatmap to visualize the pairwise p-values and decision results.

Usage

```
heat_sucess(seq_pair, test)
```

Parameters

- `seq_pair` - data frame column or character vector that contains the sequence pairs
- `test_pv`- data frame column or numeric vector that contains the p-values

edges_rej()

Function for mapping the pairwise test results on the tree. Returns a dataframe with edge id (from the ggtree object) in the first column and frequency of rejections in the second.

Usage

```
edjes_rejected_freq(tree, seq_pair, test_pv)
```

Parameters

- `tree`- ggtree object
- `seq_pairs` - data frame column or character vector that contains the sequence pairs
- `test_pv`- data frame column or numeric vector that contains the p-values

get_longest_path()

Function for getting the longest path in a tree.

Usage

```
get_longest_path(tree)
```

Parameters

- `tree`- ggtree object

compressed_tree()

Function for computing a compressed tree from the plot resulting from **coloured_tree()**. It collapses those nodes, whose immediate children have a rejection frequency lower than 0.3 and are sufficiently away from the root (30% of the length of the longest path). Additionally saves the collapsed nodes in a list.

Usage

```
compressed_tree(tree, col_tree, freq_threshold, min_root_distance)
```

Parameters

- `tree`- ggtree object
- `col_tree` - the object returned by the **coloured_tree()** function
- `freq_threshold`- the frequency based on which to collapse the trees
- `min_root_distance` - the minimum distance between the root and the collapsed nodes

References

[Gubela, 2022] Gubela, N. (2022). A test for reversibility of markov chains in molecular evolution.