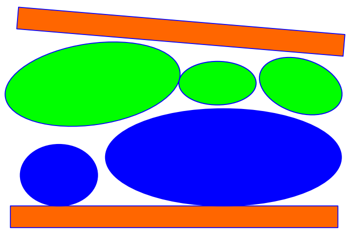
SMORE

Synteny Modulator Of Repetitive Elements

Bioinformatics Leipzig

Sarah J. Berkemer, Anne Hoffmann, Cameron R. Murray, Peter F. Stadler

27.09.2017



Contents

1 Introduction

`SMORE` is a pipeline to detect evolutionary events along a phylogenetic tree based on a set of repetitive elements and multiple sequence alignments. This manual provides a more detailed description of the `SMORE` pipeline published in [1] and [2].
An example is shown in Section 6.

The pipeline is composed of two modular parts: (1) the inference of the orthology relation and (2) the quantitative analysis of the orthology relation, Fig. 1. The first component identifies a map of genomic anchor points that are used to partition the annotated elements of interest into an initial set of candidate clusters. These are then processed to account for the most common artefacts in the input data and refined using information that is provided by analysing related but distinguishable sequence elements together. The second part of the pipeline is largely independent of the first and can also be employed using input data generated by other, third-party methods. With our pipeline, we provide an uninterrupted workflow that returns results based on input files and user-defined parameters. With the exception of breaks between subcommands indicated in Fig. 1 and where output data is provided for the user, the UNIX pipes utilizes to transfer data between software components.

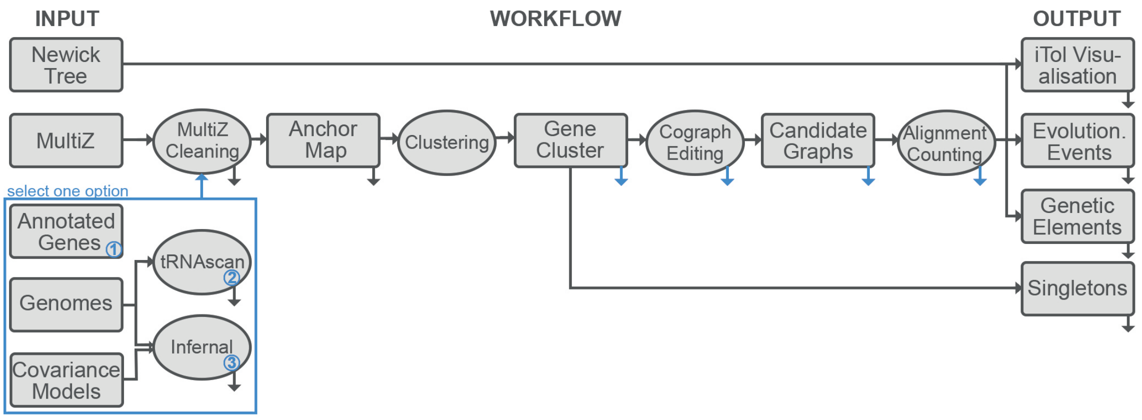


Figure 1: Summary of the computational workflow implemented in the `SMORE` pipeline for analyzing the evolution of mutlicopy genes. The compilation of orthology estimates and the quantitative analysis are logically separated and can also be used independent of each other. See text for details. Black arrows pointing into the direction of the pipeline show an uninterrupted workflow and hence no printing and reading of files in between single steps of the pipeline. Black arrows pointing downwards indicate output files that are always part of the output whereas blue arrows pointing downwards indicate the creation of temporary files and of optional output for the user, respectively.

2 Quickstart

The program can be downloaded here `https://github.com/AnneHoffmann/Smore`. No installation is required to run the SMORE pipeline except when used with additional programs such as `infernal` or `tRNAscan-SE` (see the following subsection for details).

A usual pipeline run consists of two steps: i) retrieving the clusters of genetic elements from the input data and ii) analysing the clusters and counting evolutionary events.

Parts of the pipeline can be used independently from each other in order to repeat some processing steps. Please see Section 4 (Subcommands) for further details.

To start a complete run of the `SMORE` pipeline, use

```
smore bake --out OUTPATH --ref REFERENCE_SPECIES
            --maf PATH_TO_MAF_FILES --genomes PATH_TO_GENOMEFOLDER
            --newick NEWICK_TREE <MODE-OPTIONS>
```

Upper case terms in the command need to be replaced by the specific information needed. Maf files are multiple sequence alignments in `MultiZ` format. All maf files should be loacted in the same folder. Newick tree is a tree in newick format. Only species that occur in the tree can be analysed. Each maf alignment is based on a reference species. Please give the species identifier of the maf reference species in the –ref option. The *mode-options* are options corresponding on the mode chosen for retrieving the input data. Further details in modes ar explained in Sec. 2.2 and Sec. 3. Following modes are available: (a) gene list mode, (b) loci list mode, (c) tRNAscan-SE mode and (d) infernal mode.

Note that species identifier have to be the same in the maf files, mode-dependent input data, newick tree and reference species!

The output will be written to the specified output directory (`--out` option).
The output contains a list of genetic clusters as well as a phylogenetic tree where nodes are labeled with number of evolutionary events. For more detailed information, see Section 5.

When giving input files and folders as input to the pipeline, please specify the complete absolute path, as files cannot be found without. Additionally, pathes to installed programs such as perl, python, infernal, tRNAscan-SE must not be specified except if they are not stored in the environment path variable.

2.1 Software Requirements

`SMORE` is programmed in `python` and `perl`, hence it requires `python3` and `perl5` and `gzip` in order to reduce the usage of memory. Additionally, it requires a current version of `infernal` [3] and `tRNAscan-SE` [4] in case genetic elements are automatically detected using a covariance model or using `tRNAscan-SE`. A current version of `tRNAscan-SE` can be downloaded at `http://eddylab.org/software.html`. And a current version of `infernal` at `http://eddylab.org/infernal/`. Our pipeline only requires the subprogram `cmsearch` of the `infernal` program suite.

2.2 Mode Overview

The `SMORE` pipeline includes several subcommands, that can be used independently depending on the current data set. Additionally, the subcommands `smore prep` and `smore bake` can be applied using different modes of preprocessing. Fig. 2 gives an overview of available modes. The modes will be explained in detail in the next section, Sec. 3 which will used the subcommand `smore bake` in order to explain the modes. All other subcommands will be explained in a later section, see Sec. 4.

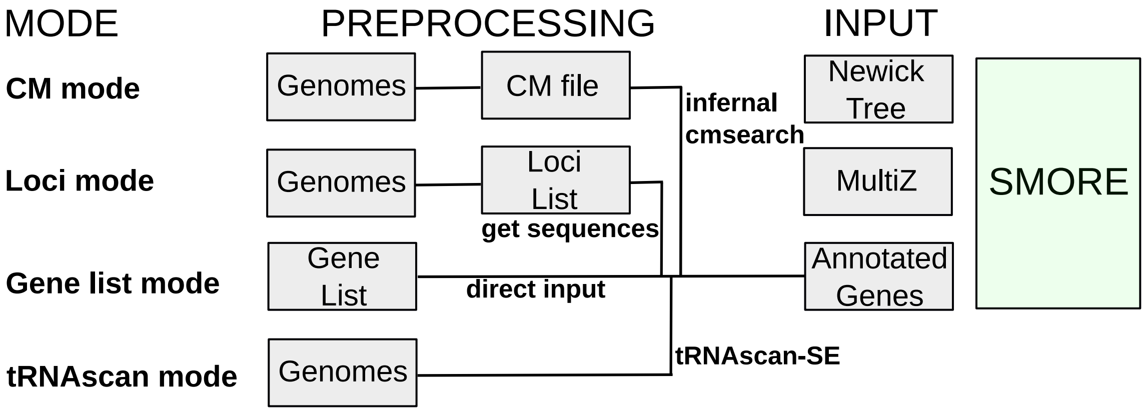


Figure 2: Overview of preprocessing modes of the `SMORE` pipeline. The preprocessing steps are part of the pipeline in order to retrieve the data needed to construct gene clusters. The construction of gene clusters will use the output of the preprocessing step which is automatically handed over.

Help pages for the pipeline can be called using one of the following commands:

```
smore --help
smore [subcommand] --help
```

3 Input for Preprocessing Modes

The subcommands `smore bake` and `smore prep` can be used running different modes of how the input gene list is created, hence different sets of parameters are required for different modes of running. This section lists all the parameters sorted by usage mode.

3.1 General input

The following parameters are used in all modes and hence, needed for subcommands `smore bake` and `smore prep`:

option	description
Obligatory parameter	
`--out|-o OUTPATH`	directory where to write the output files. If the folder does not exist, it will be created.
Optional parameter	
`--tool|-t TOOLPATH`	directory where `smore` is located. This parameter can usually be omitted.
`--python PYTHON_PATH`	Path to `python` installation if the one specified in the path environment is not used.
`--perl PERL_PATH`	Path to `perl` installation if the one specified in the path environment is not used.
`--filter NUM`	percentage of low scoring maf blocks to be discarded, NUM is between 0 and 1. This parameter is optional, default = 0.

3.2 Gene List

This mode does not run any preprocessing step but the user provided input will be directly used. This mode is used if only specific genes are included in the analysis that cannot be automatically retrieved by `tRNAscan-SE` or `infernal`.

option	description
`--genes GENELIST`	a list of genetic elements as input.

Table 3 specifies the format of the input gene list. The gene list should be tab-separated with one entry for each gene and columns should contain the information as shown in the second row of Table 3. The columns are specified by

1) chr: chromosome,

2) start: start coordinate of genetic element in the chromosome,

3) end: end coordinate of genetic element in the chromosome,

4) spec: species identifier,

5) strand: + or -,

6) type: type of genetic element e.g. Met when tRNA-Met,

7) pseudogene or not: true or false,

8) struc: secondary structure in dot-bracket notation,

9) seq: sequence,

10) comment: any comment by the user.

The third row of the table specifies if a column is obligatory (obl) or can be omitted (opt). If information in optional columns are omitted, the field has to be filled with 'NA'. For the case of optional sequence field in the gene list, please see the following subsection (loci list mode).

1	2	3	4	5	6	7	8	9	10
chr	start	end	spec	strand	type	pseudogene	struc	seq	comment
obl	obl	obl	obl	obl	opt	opt	obl	opt	opt

Table 3: This table defines the format of the gene list required as a user provided input when running smore in genelist mode. The second row provides information on which column is obligatory (obl) and which information is optional (opt). For optional information that is not provided, fields cannot be empty but have to be filled by 'NA'. Abbrevations are: chr: chromosome, spec: species, seq: sequence, struc: secondary structure.

Note that species identifier and chromosome identifier have to be the same in the maf files, gene list, newick tree and reference species!

3.3 Loci List

The loci list mode can be applied if only coordinates of elements are known. Given the corresponding genomes, `smore` will automatically retrieve the gene sequences that are needed for the analysis. Retrieving the sequences based on the provided coordinates and genomes is the only preprocessing done by the loci list mode.

option	description
`--genomes|-g GENOMES FOLDER`	folder with genomes of the species used to retrieve the sequence of the genetic elements in the loci list based on their coordinates. Filenames should match species names in loci list files.
`--loci FILE`	list with genetic elements without sequence

The loci list is a tab-separated table with one entry for each gene. The format is the same as for the gene list but omits the last three columns, as shown in Table. 5. The columns are specified as follows:

1) chr: chromosome,

2) start: start coordinate of genetic element in the chromosome,

3) end: end coordinate of genetic element in the chromosome,

4) spec: species identifier,

5) strand: + or -,

6) type: type of genetic element e.g. Met when tRNA-Met,

7) pseudogene or not: true or false,

The third row of the table specifies if a column is obligatory (obl) or can be omitted (opt). If information in optional columns are omitted, the field has to be filled with 'NA'.

1	2	3	4	5	6	7
chr	start	end	spec	strand	type	pseudogene
obl	obl	obl	obl	obl	opt	opt

Table 5: This table defines the format of the gene list required as a user provided input when running smore in genelist mode. The second row provides information on which column is obligatory (obl) and which information is optional (opt). For optional information that is not provided, fields cannot be empty but have to be filled by 'NA'. Abbrevations are: chr: chromosome, spec: species.

Note that species identifier and chromosome identifier have to be the same in the maf files, genome files, loci list, newick tree and reference species!

3.4 Infernal

The `infernal` program suite [3] provides several programs to search for genes with a specified covariance model (CM), to scan given genomes or create covariance model for a set of training sequences. Our pipeline uses `cmsearch` in order to retrieve from a given genome based on a provided CM. Hence, `infernal` can be used to create the CM in advance. For more details, we refer to the `infernal` manual or publication [3].

option	description
`--cm|-c CM`	covariance model file, input for infernal
`--genomes|-g GENOMES FOLDER`	folder with genomes of the species used as an input to infernal to scan the genomes for genetic elements specified by the CM. Filenames should match species names in MultiZ files.
`--incE NUM`	optional parameter for cmsearch (infernal), e-value threshold.
`--incT NUM`	optional parameter for cmsearch (infernal), bitscore threshold.
`--infernal PATH`	path to cmsearch, only needed if the one specified in the path environment is not used.
`--pseudo NUM`	optional parameter, bitscore threshold that defines pseudogenes.

Note that species identifier and chromosome identifier have to be the same in the maf files, genome files, newick tree and reference species!

3.5 tRNAscan mode

In bioinformatics applications, `tRNAscan-SE` [4] is a popular tool to retrieve tRNAs from a given genome. Hence, when in tRNAscan mode, our pipeline automatically applies `tRNAscan-SE` on the given genomes. The `tRNAscan-SE` output files will be parsed and reformatted in order to retrieve all information needed to construct gene clusters. As mentioned in [2], tRNA genes are the best studied elements when working on concerted evolution. tRNA genes show a high turnover rate and frequent pseudogenization events.

option	description

`--genomes|-g GENOMES FOLDER`	folder with genomes of the species used as an input to infernal to scan the genomes for genetic elements specified by the CM. Filenames should match species names in MultiZ files.
`--trna`	option to activate the usage of tRNAscan-SE on the given genomes.
`--trnascan PATH`	path to tRNAscan-SE, only needed if it is not installed in the environment path variable.

Note that species identifier and chromosome identifier have to be the same in the maf files, genome files, newick tree and reference species!

4 Subcommands

In order to repeat only some parts of the pipeline or have a more detailed look into intermediary files, parts of the pipeline can be used independently. This section gives an overview of all possible subcommands explaining input, output and options. Fig. 3 gives a graphical overview of subcommands in comparison to the workflow.
The `SMORE` pipeline can be applied in one complete run but also splitted in several parts that allow the user to change and compare results created by different parameter sets. Additionally, the pipeline of subcommnands consisting of `smore prep, smore mix, smore roast, smore eat` is able to process large amounts of data by splitting the set of clusters in disjoint subsets (`smore mix`), analyse each part individually (`smore roast`) and summarizing all results at the end (`smore eat`).

4.1 `smore bake`

The subcommand `smore bake` will run the complete pipeline for a given input. Different modes such as gene list or infernal mode can be specified by the set of parameters and input data given. The output is saved in the specified output folder (option `--out`). In case the output folder does not exist, it will be created. As `smore bake` is a combination of the subcommands `prep` and `toast`, the parameter set is the same as for `prep` and `toast`. Thus, please see the tables in the next two subsections for parameter descriptions. The help pages for `smore bake` is `smore bake --help` or `smore bake -h`.

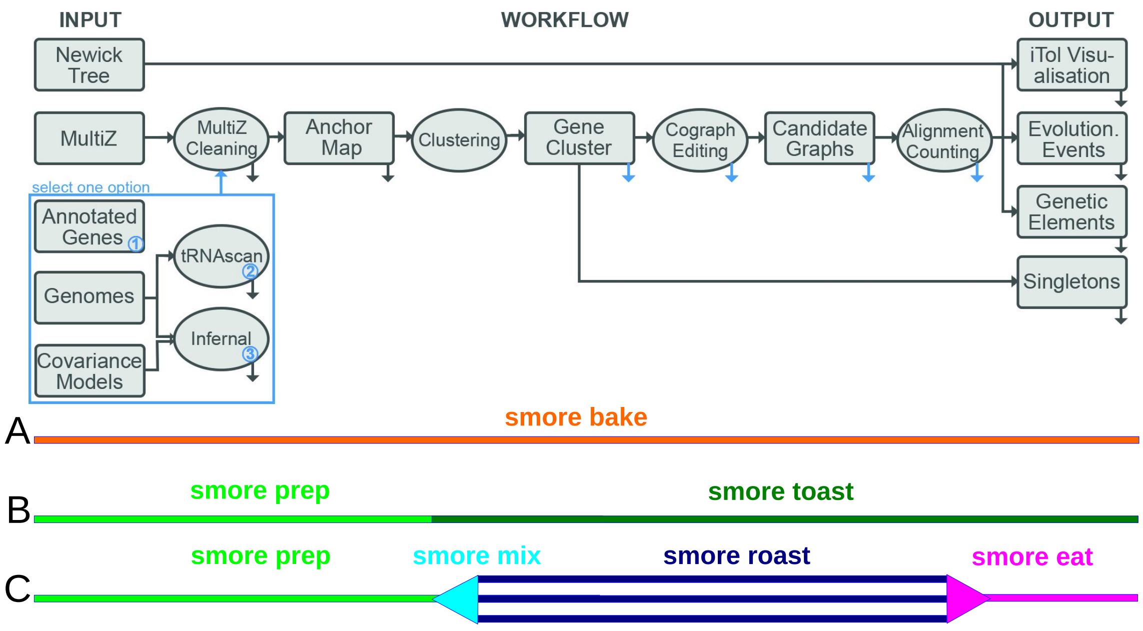


Figure 3: Overview of subcommands of the `SMORE` pipeline next to the workflow.

4.2 `smore prep`

By applying `smore prep` only the first part of the pipeline will be executed. Hence, only input data concerning the creation of genetic clusters is required. This subcommand can be used to create a basic dataset that will be the input of following steps of the pipeline. Afterwards, succeeding subcommands can be applied with different parameter sets. Depending on the number of genetic elements and genome size, `smore prep` might take several hours to process the input data. The output will be saved in the specified output folder (option `--out`). The help page for `smore prep` is `smore prep --help` or `smore prep -h`.

parameter	obl/opt	description
General		
`--out|-o OUTPATH`	obl	directory where to write the output files. If the folder does not exist, it will be created.
`--maf MAF_FILES`	obl	directory where all multiZ alignment files are located.
`--ref REFERENCE_SPECIES`	obl	name of the reference species as specified in the multiZ alignment files.
`--tool|-t TOOLPATH`	opt	directory where *SMORE* is located. This parameter can usually be omitted.

`--python PYTHON_PATH`	opt	Path to `python` installation if the one specified in the path environment is not used.
`--perl PERL_PATH`	opt	Path to `perl` installation if the one specified in the path environment is not used.
`--filter NUM`	opt	percentage of low scoring maf blocks to be discarded, NUM is between 0 and 1. This parameter is optional, default = 0.
Genelist Mode		
`--genes GENELIST`	obl	a list of genetic elements as input.
Infernal mode		
`--cm|-c CM`	obl	covariance model file, input for infernal
`--genomes|-g GENOMES FOLDER`	obl	folder with genomes of the species used as an input to infernal to scan the genomes for genetic elements specified by the CM. Filenames should match species names in MultiZ files.
`--incE NUM`	opt	parameter for cmsearch (infernal), e-value threshold.
`--incT NUM`	opt	parameter for cmsearch (infernal), bitscore threshold.
`--infernal PATH`	opt	path to cmsearch, only needed if the one specified in the path environment is not used.
`--pseudo NUM`	opt	bitscore threshold that defines pseudogenes.
tRNAscan-SE mode		
`--genomes|-g GENOMES FOLDER`	obl	folder with genomes of the species used as an input to infernal to scan the genomes for genetic elements specified by the CM. Filenames should match species names in MultiZ files.
`--trna`	obl	option to activate the usage of tRNAscan-SE on the given genomes.
`--trnascan PATH`	opt	path to tRNAscan-SE, only needed if it is not installed in the environment path variable.
Locilist mode		

`--genomes|-g` `GENOMES FOLDER`	obl	folder with genomes of the species used to retrieve the sequence of the genetic elements in the loci list based on their coordinates. Filenames should match species names in loci list files.
`--loci FILE`	obl	list with genetic elements without sequence

4.3 `smore toast`

The subcommand `smore toast` can be applied after `smore prep`. It will take `smore prep` output (option `--prep`) and analyse the given genetic clusters. The output will be saved in the specified output folder (option `--out`). As `smore toast` can be applied using different sets of parameters, applying it several times on the same input data can help estimating the best parameter set e.g. joining of clusters (`--join`) or similarity thresholds (`--seq, --struc`). By omitting verbose output, `smore toast` is able to process given input data in only several minutes. Verbose output (if `--verbose`) or any combination of them (if `--clus, --graph, --aln`) will help to get a deeper insight into intermediary steps and how the clusters are composed but will take a lot more time to process the data and more space capacities as all intermediary files are stored in the output folder. The help page for `smore toast` is `smore toast --help`.

parameter	obl/opt	description
`--out|-o OUTPATH`	obl	directory where to write the output files. If the folder does not exist, it will be created.
`--prep PATH`	obl	directory where `smore prep` output files are located.
`--newick NEWICK_FILE`	obl	phylogenetic tree in newick format including all species contained in the analysis. For automatic conversion of species identifiers, use parameter –id.
`--id ID_FILE`	opt	in case species identifier in the tree and multiZ files do not match, this file can be used to automatically convert species identifier. Format of the table: current_name_in_tree (tab) name_to_translate_to

`--join LEVEL`	opt	level of how to join the clusters. Levels are *none, strict* and *relaxed.* Default: *relaxed.* The level refers to the adjacency constraints of gene clusters, see Sec. 7 for further details.
`--seqsim NUM`	opt	percentage of sequence similarity in order to define two sequences as orthologs. Number between 0 and 1, default: 0.9. Set to -1 in case the sequence similarity should be omitted for the analysis (hence, only check structure similarity).
`--strucsim NUM`	opt	percentage of structure similarity in order to define two structures as orthologs. Number between 0 and 1, default: 0.9 Set to -1 in case the structure similarity should be omitted for the analysis (hence, only check sequence similarity).
`--nomiss`	opt	using this option, the pipeline will omit the check for missing data. Hence, deletions in the tree won't distinguish between a deletion because of missing data or a 'real' deletion. Only useful when running on a large data set.
`--verbose`	opt	using this option, the pipeline will print all intermediary files such as clusters, graphs and alignments. This will slow down the running time of the program but will give deeper insights into the data.
`--clus`	opt	using this option, the pipeline will print all intermediary cluster files
`--graph`	opt	using this option, the pipeline will print all intermediary graph files
`--aln`	opt	using this option, the pipeline will print all intermediary alignment files
`--tool|-t TOOLPATH`	opt	directory where *SMORE* is located. This parameter can usually be omitted.
`--python PYTHON_PATH`	opt	Path to `python` installation if the one specified in the path environment is not used.

`--perl PERL_PATH`	opt	Path to `perl` installation if the one specified in the path environment is not used.

4.4 `smore mix`

`smore mix` is designed to only run the joining of the clusters based on given parameters and `smore prep` output. It will return a list of the original clusters and another file listing clusters after joining. Additionally, `smore mix` can be used to handle large amounts of data. With option `--max` the maximal number of clusters can be specified that should be included in the pipeline's next processing step. `smore mix` will hence split the resulting clusters in several files that do not exceed the specified number. In order to help the user proceed to the next step of the pipeline (`smore roast`), the output will contain a command list, specifying the commands needed to call the next step with each part of the cluster list. The user can easily copy the commands and call the next step or modify the commands by adding parameters. Verbose output can be used with `smore mix` but might slow down running time of the program. The possible parameters for `smore mix` include parameters for `smore roast`, too, as the parameters will be used to create the commandlist. The help page for `smore mix` is `smore mix --help`.

parameter	obl/opt	description
`--out|-o OUTPATH`	obl	directory where to write the output files. If the folder does not exist, it will be created.
`--prep PATH`	obl	directory where `smore prep` output files are located.
`--species FILE`	obl	file listing all species included in the analysis. Thus, the file includes the species identifier, one per line. The file is created by `smore prep` and thus will be located in the `smore prep` output folder, named `specieslist`.
`--newick NEWICK_FILE`	obl	phylogenetic tree in newick format including all species contained in the analysis. For automatic conversion of species identifiers, use parameter –id.

`--id ID_FILE`	opt	in case species identifier in the tree and multiZ files do not match, this file can be used to automatically convert species identifier. Format of the table: current_name_in_tree (tab) name_to_translate_to
`--join LEVEL`	opt	level of how to join the clusters. Levels are *none, strict* and *relaxed*. Default: *relaxed*. The level refers to the adjacency constraints of gene clusters, see Sec. 7 for further details.
`--max NUM`	opt	this parameter can be used to specify the maximal number of clusters used in following steps of the analysis. If there are more clusters, the program will automatically split the data set and create a command list in order to run the next steps in parallel. The default value is 50000.
`--seqsim NUM`	opt	percentage of sequence similarity in order to define two sequences as orthologs. Number between 0 and 1, default: 0.9. Set to -1 in case the sequence similarity should be omitted for the analysis (hence, only check structure similarity).
`--strucsim NUM`	opt	percentage of structure similarity in order to define two structures as orthologs. Number between 0 and 1, default: 0.9. Set to -1 in case the structure similarity should be omitted for the analysis (hence, only check sequence similarity).
`--nomiss`	opt	using this option, the pipeline will omit the check for missing data. Hence, deletions in the tree won't distinguish between a deletion because of missing data or a 'real' deletion. Only useful when running on a large data set.
`--verbose`	opt	using this option, the pipeline will print all intermediary files such as clusters, graphs and alignments. This will slow down the running time of the program but will give deeper insights into the data.

`--clus`	opt	using this option, the pipeline will print all intermediary cluster files
`--tool|-t TOOLPATH`	opt	directory where *SMORE* is located. This parameter can usually be omitted.
`--python PYTHON_PATH`	opt	Path to `python` installation if the one specified in the path environment is not used.
`--perl PERL_PATH`	opt	Path to `perl` installation if the one specified in the path environment is not used.

4.5 `smore roast`

The subcommand `smore roast` is applied on the `smore mix` output. The program `smore mix` will output a list of commands on how to proceed. Thus `smore roast` is called based on these commands. However the user can change or add some parameters to the preprinted commands in order to adjust the program call. If there is more than one commands, it is important that all commands are executed with the same parameter set in order to obtain a homogenuous result at the end. If some of the commands are omitted, the final result will only include the data that was analysed. The next step of the pipeline `smore eat` will summarize the `smore roast` output into final results of the pipeline run. The resulting files of `smore roast` are given specified names to make sure that the next step of the pipeline will find all input data. Thus, file names should not be changed and files not be moved to another folder. Verbose output can be used with `smore roast`, also for just some of the commands but might slow down the running time of the program. The help page for `smore roast` is `smore roast --help`.

parameter	obl/opt	description
`--out|-o OUTPATH`	obl	directory where to write the output files. If the folder does not exist, it will be created.
`--in|-i FILE`	obl	Output file of `smore mix` including the list of clusters.
`--species FILE`	obl	file listing all species included in the analysis. Thus, the file includes the species names as given in the newick tree and prep output, one species identifier per line.

`--newick NEWICK_FILE`	obl	phylogenetic tree in newick format including all species contained in the analysis. For automatic conversion of species identifiers, use parameter –id.
`--id ID_FILE`	opt	in case species identifier in the tree and multiZ files do not match, this file can be used to automatically convert species identifier. Format of the table: current_name_in_tree (tab) name_to_translate_to
`--seqsim NUM`	opt	percentage of sequence similarity in order to define two sequences as orthologs. Number between 0 and 1, default: 0.9. Set to -1 in case the sequence similarity should be omitted for the analysis (hence, only check structure similarity).
`--strucsim NUM`	opt	percentage of structure similarity in order to define two structures as orthologs. Number between 0 and 1, default: 0.9. Set to -1 in case the structure similarity should be omitted for the analysis (hence, only check sequence similarity).
`--nomiss`	opt	using this option, the pipeline will omit the check for missing data. Hence, deletions in the tree won't distinguish between a deletion because of missing data or a 'real' deletion. Only useful when running on a large data set.
`--verbose`	opt	using this option, the pipeline will print all intermediary files such as clusters, graphs and alignments. This will slow down the running time of the program but will give deeper insights into the data.
`--graph`	opt	using this option, the pipeline will print all intermediary graph files
`--aln`	opt	using this option, the pipeline will print all intermediary duplication alignment files
`--tool|-t TOOLPATH`	opt	directory where *SMORE* is located. This parameter can usually be omitted.

`--python PYTHON_PATH`	opt	Path to `python` installation if the one specified in the path environment is not used.
`--perl PERL_PATH`	opt	Path to `perl` installation if the one specified in the path environment is not used.

4.6 `smore eat`

The subcommand `smore eat` is applied on output of `smore roast`. Here, the output folder of `smore roast` is the input to `smore eat` which will take into account all files in the folder with having specific names as given by `smore roast`. This last part of the program will then summarize all the information such that evolutionary events can be counted and numbers added to the tree. The help page for `smore eat` is `smore eat --help`.

parameter	obl/opt	description
`--out|-o OUTPATH`	obl	directory where to write the output files. If the folder does not exist, it will be created.
`--prep PATH`	obl	directory where `smore prep` output files are located.
`--mix PATH`	obl	directory where `smore mix` output files are located.
`--roast PATH`	obl	directory where `smore roast` output files are located.
`--newick NEWICK_FILE`	obl	phylogenetic tree in newick format including all species contained in the analysis. For automatic conversion of species identifiers, use parameter –id.
`--id ID_FILE`	opt	in case species identifier in the tree and multiZ files do not match, this file can be used to automatically convert species identifier. Format of the table: current_name_in_tree (tab) name_to_translate_to
`--nomiss`	opt	using this option, the pipeline will omit the check for missing data. Hence, deletions in the tree won't distinguish between a deletion because of missing data or a 'real' deletion. Only useful when running on a large data set.
`--tool|-t TOOLPATH`	opt	directory where *SMORE* is located. This parameter can usually be omitted.

`--python PYTHON_PATH`	opt	Path to `python` installation if the one specified in the path environment is not used.
`--perl PERL_PATH`	opt	Path to `perl` installation if the one specified in the path environment is not used.

5 Output

All output files are located in the specified folder, given with option `--out` or `-o`. The output consists of multiple files and depends on the subcommand(s) used. The following subsections describe the output files in more details. For an example, we refer to the next section, Sec. 6.

5.1 General Output

Each of the `smore` subcommands will provide an output file that summarizes the current run and provides information about the status of the program during runtime. Additionally, each subpart of the pipeline will create a file where all errors are collected. If the file is empty, no errors occured.

output file	description
Summary.txt	This file gives a short overview of the program's run specifying parameter and running time. This file can also be used to check the status of the program while running as it gives information based on the current steps of the pipeline.
errors_SUBCOMMAND	for each part of the smore pipeline, there is a file giving errors that happened during the run. If no errors occured, the file is empty.

5.2 Output of completed pipeline run

As specified in Sec. 2.2 by Fig. 2, there are several ways on how to do a complete run of the `smore` pipeline. The options are:

1. `smore bake`
2. `smore prep` and `smore toast`

3. `smore prep` and `smore mix` and `smore roast` and `smore eat`

This section will describe the output after running one of the the three options mentioned above. For output of intermediary steps, please see one of the following subsections, depending on the subcommand. The table lists the output files including final results. For intermediary files, see following tables.

output file	description
OutTree.txt	the resulting tree in newick format with numbers at the nodes given in brackets. This format can be used to visualize the tree with newick compatible programs.
geneticEvents.txt	File listing all genetic events counted during the analysis. The numbers are sorted by event and node of the tree. The file includes a event called 'Other'. This will give the difference of genetic elements between the total amount and the elements used in the analysis. For a successful run of the pipeline, the numbers should be 0.
data_iTOL	folder giving files that can be uploaded to *itol.embl.de* [5]. The file called F0tree.txt is uploaded at http://itol.embl.de/upload.cgi. The remaining files can be added using drag and drop into the browser window. This will result in a interactive visualization of the resulting tree. A legend is added automatically. All nodes in the tree will have unique names, thus some nodes might have names such as 'innerNode0' because it was added automatically. For an explanation of how to visualize the final output, see Sec. 5.7.

allClusters_original.txt, allClusters_joined.txt	These two files contain lists of clusters showing which elements are contained together in one cluster before and after joining. A line starting with '¿' is followed by the cluster number. Each line after the cluster ID contains one genetic element of the cluster having the following format: `chromosome species_elementID start end strand left_block right_block structure sequence type pseudogene comment`. In case some information is not provided, the field is filled with 'NA'.
list_cographs.txt, list_noncographs.txt	these files contain statistics about graphs that were cographs from the beginning or had to be edited in order to become a cograph. The tables list number of nodes, number of edges, number of corrected edges (see Sec. 7) and density of the graphs.
remoldings.txt, inremoldings.txt	These files contain genetic elements that (a) have highly similar sequences but different types or (b) have the same types but clearly distinct sequences.
allTypes.txt, allPseudoTypes.txt	These two files list the different types of genetic elements for all species and functional or pseudogenized genes. This can be used to analyse the distribution of different types of genetic elements.

5.3 Verbose Output

the verbose version of toast will output three additional files (if `--verbose`) or any combination of them (if `--clus, --graph, --aln`) for each cluster, named with left and right anchor numbers to match all three files. They will be in three different folders: cluster, graph and duplication_alignment. The files contain the specific structures of the cluster in each step of the analysis and can be used to gain a deeper insight into the data.

output file	description

GENECLUSTER.clus	the files consist of a list of genetic elements being part of this cluster. There is one entry for each genetic element in the cluster. The format for an entry is a tab-separated line with following fields: `chromosome species_elementID start end strand left_block right_block structure sequence type pseudogene comment`. In case some information is not provided, the field is filled with 'NA'.
GENECLUSTER.edli	files showing the graph structure of a gene cluster using a weighted edge list. The graphs are complete, thus there exists an edge between each pair of nodes. Hence, there is one entry for each edge in the graph. The entry is a tab-separated line with format: `node1 node2 seq_sim struc_sim`. The nodes are genetic elements and the weights are the sequence similarity score and structure similarity score (if structure is available).
GENECLUSTER.aln	The alignment files show the results of the duplication alignments for each cluster. Hence, for each species in the cluster, there is a sequence of genetic elements. Those elements are represented by single letters. If the elements' similarity is above the thresholds, they share the same letter. The duplication alignment are a Needleman-Wunsch alignment including the possibility to detect duplications. Duplications are represented by ' ', gaps by '-'. For each pair of species in a cluster, there is an entry consisting of the alignment and the alignment score.

5.4 Output `smore prep`

The subcommand `smore prep` is the first part of the pipeline, sorting genetic elements in between genomic anchors in order to create gene clusters. Hence, `smore prep` will output data in three different subfolders in the output folder:

1. **genes**: This data is needed to continue the analysis of gene clusters. The folder contains a file in bed format for each of the species. The files contain a tab-saparated entry line for each genetic element. The format is as follows: `chromosome species_elementID start end strand left_anchorID right_anchorID structure sequence type pseudogene comment`. In case some information is not provided, the field is filled with 'NA'.

2. **bed**: This data is not used any further in the analysis but might be useful when having a deeper look into the results. The folder contains a gzipped file for each species. The files contain information about the genomic anchors retrieved from the MultiZ files. For each genomic anchor, there is one tab-separated line with the following format: `chromosome species_anchorID start end strand ID_of_leftadjacent_anchor ID_of_rightadjacent_anchor`. In case no anchor could be detected, the field is filled with 'None'.

3. **temp**: The files in this folder are needed in a later step of the analysis when deletions in the tree are clustered by cause, hence either missing data or deletion of the genetic elements. The folder contains a gzipped file for each species containing one entry for each genomic anchor. The information differs from the one in the bed folder as it contains scores for each of the anchors. The format is a tab-separated line for each anchor with `chromosome species_anchorID anchorStart anchorLength strand refSpecies anchorScore`. The 'refSpecies' field will contain 'True' if the species is the reference species, and 'False' if not.

5.5 Output `smore mix`

The subcommand `smore mix` reads `smore prep` output as its input and summarizes gene clusters. The program will join clusters based on the specified parameter (`--join`) and in case the number of cluster is higher than the specified maximal number (`--max`), it will split the set of clusters. In this way, the next step of the pipeline can be started on several subsets of clusters which will speed up the overall running time. Hence, `smore mix` will create a folder called *clusterlists* containing list of gene clusters. If all the lists are combined, the original set of clusters is retrieved. Additionally, `smore mix` will output a file called *commandlist.txt*. This file consists of a list of commands calling `smore roast` for each of the clusterlists. Hence, the user only needs to copy-paste the commands to execute or add or edit some parameters before running the commands. Additionally, `smore mix` creates output files listed in the following table.

output file	description
specieslist	This files contains the species' identifier as they appear in the analysis, one line for each species.
nones.txt, pseunones.txt	Each genetic element that cannot be set in between genomic anchors, the anchor identifier is specified as 'None'. After starting the pipeline, all genetic elements with 'None'-anchors will be excluded from the analsis and written in a separate file.

5.6 Output `smore roast`

The subcommand `smore roast` will use `smore mix` output, hence a clusterlist. In case, `smore roast` is applied in parallel to several subdivided parts of the original list of clusters, `smore roast` will output intermediary files for each run that can be summarized running `smore eat`. Hence, `smore roast` output files should stay in the specified folder and not be renamed or moved. The follwing table specifies the output files. The name of the corresponding clusterlist will be added to the filenames in order to have unique names in case `smore roast` is started several times. For further explanations on event counts, see Sec. 7.

output file	description
singletons, pseusingletons	Lists counts for singletons or pseudogene singletons (if present) for each species.
matches, pseudomatches	Lists counts for matches or pseudogene matches (if present) for each species.
insertions, pseudoinsertions	Lists counts for insertions or pseudogene insertions (if present) for each species.
duplications	Lists counts for duplications for each species.
delCheck, pseudelCheck	Lists counts for deletions or pseudogene deletions (if present) for each species. The deletions might be caused due to missing data or evolutionary deletion of genes. This will be checked during `smore eat`.
remoldings, inremoldings	Lists counts for remoldings or inremoldings (if present) for each species.

errors	For each run of `smore roast` there will be a file listing errors occured during the run. If the file is empty, no errors occured.
list_cographs, list_noncographs	Lists cographs and non-cographs with identifier, node number, edge number, corrected edge number and density for each graph.

5.7 Output Visualization

After a complete successful run of the pipeline, the specified output folder will contain a subfolder called *data_iTOL*. This folder contains files that are formatted in a way such that after uploading them on the `iTOL` webpage, the resulting phylogenetic tree will be displayed in an interactive environment. Hence, the tree can be edited further and downloaded in several formats. The filenames are numbered, F0 to F8, which shows the recommended order in which they are uploaded. Changing the order does not cause any problems, except for file F0, which needs to be uploaded first, as it contains the underlying tree structure.

The following steps have to be done to visualize the tree:

1. after the folder *data_iTOL* is created and contains files F0 to F8, open the `iTOL` webpage in a browser, hence either open `http://itol.embl.de` and click on *Annotate* or directly open `http://itol.embl.de/upload.cgi`.
2. Click on *Browse* and choose the file *F0tree.txt* from the folder *data_iTOL*. After clicking on *Upload* the tree structure is displayed in the browser.
3. the remaining files from the folder *data_iTOL* can be added to the tree by using drag and drop on the files. Following the order specified by the numbering of the files is recommended but not mandatory.
4. Using the interactive browser tool `iTOL`, colors and labels can be changed, as well as sizes of branches and distances between nodes. The tool provides several formats for downloading the tree.
5. Numbers in the tree are either written at the leaves or at inner edges of the tree. Numbers at the leaves are specific for the species whereas numbers at inner edges refer to all species that are 'below' of 'after' that edge on the way from the root to

the leaves. Numbers are color-coded and a legend is provided. Colors can be changed using `iTOL`.

6. In order to display the tree correctly, every node will have a label. If the label was not present in the original tree, the nodes will receive a name during the pipeline's run. Hence, some nodes will have identifier such as 'InnerNode0'. These can of course be deleted.

7. For examples on how the trees look like, please see Sec. 6.

6 Examples

This section will show some examples of how to start the pipeline and how to read and visualize the output files.

6.1 `smore bake` with `tRNAscan-SE`

The `smore` pipeline will be started using 6 primate species and tRNAscan-SE in order to obtain counts of evolutionary events for tRNAs withing primate species. The species identifier have to be the same for the genomes, MultiZ files, reference species and in the newick tree. The data of this example was published in [2]. The input data such as MultiZ files were downloaded from `http://hgdownload.cse.ucsc.edu/downloads.html`. Genomes in fasta format and phylogenetic tree in newick format can be downloaded from `UCSC` or `NCBI`. Table 18 lists the species included in the analysis.

When giving input files and folders as input to the pipeline, please specify the complete absolute path, as files cannot be found without. Additionally, pathes to installed programs such as perl, python, infernal, tRNAscan-SE must not be specified except if they are not stored in the environment path variable.

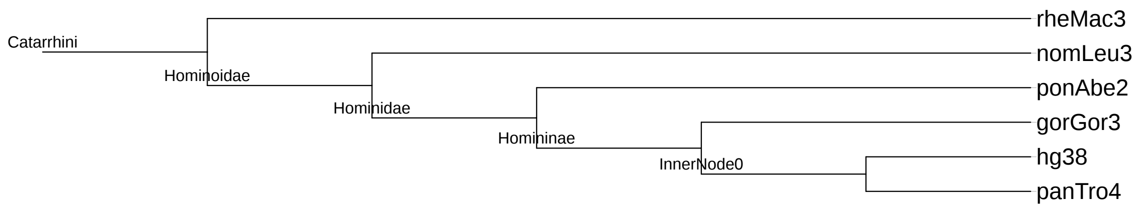


Figure 4: Phylogenetic tree displayed by `iTOL` as given as an input to `smore bake`. As for the analysis, all nodes have to have labels, the program will automatically name nodes without label. Hence, one of the inner nodes in the tree is called *InnerNode0*.

species	abbreviation
Homo sapiens	hg38
Pan troglodytes	panTro4
Gorilla gorilla	gorGor3
Pongo abelii	ponAbe2
Nomascus leucogenys	nomLeu3
Macaca mulatta	rheMac3

Table 18: Species used in the analysis with scientific name and abbreviation used in our analysis.

The pipeline is called as follows. Uppercase names have to be changed based on the system where the pipeline is executed.

```
smore bake --out OUTPATH/Output_smorebake --ref hg38
            --maf PATH_TO_MAF_FILES --genomes PATH_TO_GENOMEFOLDER
            --newick tree_6primates.newick --trna
```

Fig. 4 shows the tree given as an input to `smore bake` displayed using `iTOL`. The newick format of the tree is *(rheMac3,(nomLeu3,(ponAbe2,((hg38,panTro4)InnerNode0,gorGor3)Homininae)Hominidae)Hominoidae)Catarrhini;*.

As described in Sec. 5.7, the interactive tool `iTOL` can be used to edit the tree and adapt features such as size, colors, labels. The resulting slightly edited tree of the `smore bake` run on 6 primate species and tRNAs is displayed in Fig. 5.

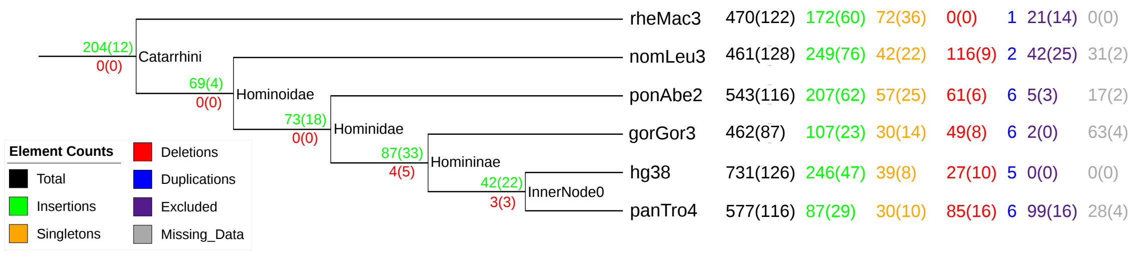

rheMac3 470(122) 172(60) 72(36) 0(0) 1 21(14) 0(0)
nomLeu3 461(128) 249(76) 42(22) 116(9) 2 42(25) 31(2)
ponAbe2 543(116) 207(62) 57(25) 61(6) 6 5(3) 17(2)
gorGor3 462(87) 107(23) 30(14) 49(8) 6 2(0) 63(4)
hg38 731(126) 246(47) 39(8) 27(10) 5 0(0) 0(0)
panTro4 577(116) 87(29) 30(10) 85(16) 6 99(16) 28(4)
204(12)
0(0)
Catarrhini
69(4)
0(0)
Hominoidae
73(18)
0(0)
Hominidae
87(33)
4(5)
Homininae
42(22)
3(3)
InnerNode0
Element Counts
Total
Insertions
Singletons
Deletions
Duplications
Excluded
Missing_Data

Figure 5: Resulting phylogenetic tree from the `smore bake` run on 6 primate species using `tRNAscan-SE` displayed by `iTOL`.

Further output files are a file called `Summary.txt` which includes status reports about the program's execution and statistics about the number of clusters, the size of clusters and information about graph structures and the number of corrected graphs.
For the current example we get the following statistics:

```
Number of species: 6
Number of different element types: 24
Number of original clusters: 1926
Average number of elements of original clusters: 1.93
Number of joined clusters: 1038
Average number of elements per cluster in joined cluster: 3.58

Information on graph structures and corrections:
Number of cographs 650, with on average
3.74 nodes, 4.78 edges and a density of 0.82.
Number of non-cographs 5, with on average
8.40 nodes, 21.80 edges and a density of 0.28.
All non-cographs were corrected to obtain a cograph structure.
```

Further output files, called `allTypes.txt` and `allPseudoTypes.txt` can be used to visualize the distribution of element types and species in the data set. For each pair of species and element type, there is one entry in the files giving the number of elements with this combinations. Fig. 6 shows the distribution of types within active and pseudogenized tRNA genes.

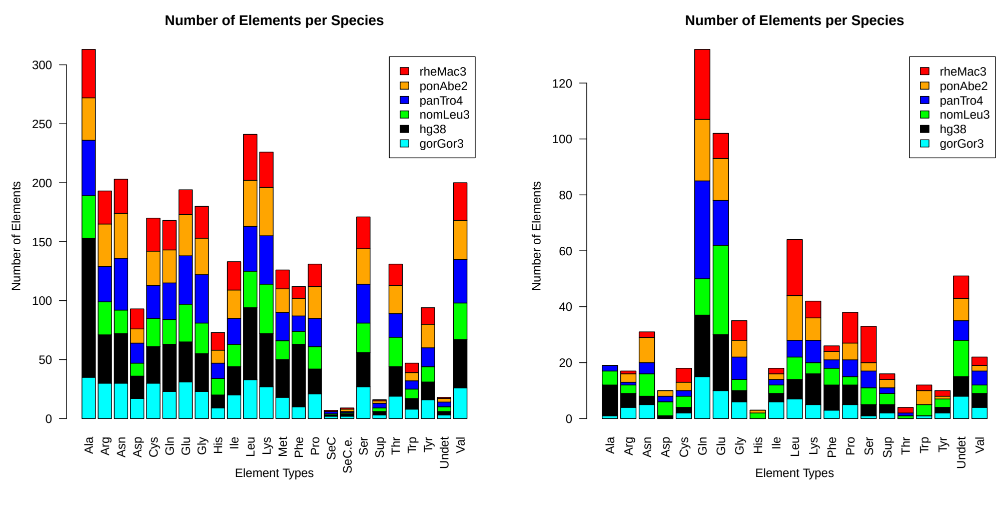


Figure 6: The plots show the distribution of tRNA types in the analysis based on 6 primate species. Plot A shows the distribution of active tRNA genes whereas Plot B shows the distribution of pseudogene types. The types of genes are identified using `tRNAscan-SE`.

6.2 `smore bake` with loci list and gene list

The calls for `smore bake` with loci list and gene list are very similar, and thus explained both here. A loci list is a list which contains information about genetic elements such as chromosome, species, strand and start and end coordinates. In order to apply similarity thresholds, our pipeline needs the genomic sequences and/or secondary structures of the elements. Hence, the loci list will be extended by the sequences taken from the species' genomes.

The call for running `smore bake` with a loci list as input looks as follows:

```
smore bake --out OUTPATH/Output_smorebake --ref hg38
             --maf MAF_FILES_FOLDER --genomes GENOMEFOLDER
             --newick tree_6primates.newick --loci locilist.tsv
```

Calling `smore bake` with a gene list has a similar call but the genomes are not needed:

```
smore bake --out OUTPATH/Output_smorebake --ref hg38
             --maf MAF_FILES_FOLDER
             --newick tree_6primates.newick --gene genelist.tsv
```

As an example, the **loci list** with tRNAs in hg38 looks as follows:

```
chr1 16725566 16725638 hg38 + Val T
chr1 16727285 16727355 hg38 + Gly F
chr1 16860198 16860270 hg38 + Val T
chr1 16861921 16861991 hg38 + Gly F
chr1 16872583 16872654 hg38 + Glu F
```

The first column specifies the chromosome, the next two columns are start and end coordinates of the gene. The fourth and fifth columns specify the species and the strand. The last two columns show the type of the gene and if the gene is considered to be a pseudogene (T) or not (F).

The following shows the same entries as before, but here as a **gene list** input (sequences and structures shortened):

```
chr1 16725566 16725638 hg38 + Val T >>>>>>.[..]<<<<<<. GTTTCTG[..]GGAAACA NA
chr1 16727285 16727355 hg38 + Gly F >.>>>>.[..]<<<.<<. GCCTTGG[..]CAATGCA NA
chr1 16860198 16860270 hg38 + Val T >>>>>>.[..]<<<<<<. GTTTCTG[..]GGGAACA NA
chr1 16861921 16861991 hg38 + Gly F >>>>>>.[..]<<<<<<. GCATTGG[..]CAATGCA NA
chr1 16872583 16872654 hg38 + Glu F >>>>>>.[..]<<<<<<. TCCCTGG[..]CAGGGAA NA
```

The gene list additionally includes a column specifying the secondary structure in dot-bracket notation and a column for the sequence. The last column of the gene list is a comment column which might be useful for user-created list that only contain very specific genes.

6.3 `smore bake` with `infernal`

In order to apply `smore bake` using the infernal option, a covariance model has to be specified using option `--cm`. The call for running `smore bake` with a loci list as input looks as follows:

```
smore bake --out OUTPATH/Output_smorebake --ref hg38
             --maf MAF_FILES_FOLDER --genomes GENOMEFOLDER
             --newick tree_6primates.newick --cm trna.cm
```

Ready-to-use covariance models can be downloaded at `http://rfam.xfam.org/`. Otherwise, covariance models can be created based on sample sequences using `infernal` [3].

6.4 `smore prep` and ..

As depicted in Fig. 3 showing the different combinations of subcommands, it is possible to run parts of the pipeline independently from each other. The first part, including the preprocessing step, sorts the genetic elements inbetween the genomic anchors and thus, defines gene clusters. This part is executed when calling `smore prep`. As explained in Sec. 5.4, `smore prep` output consists of three different folders. This data forms the basis for further analysis steps using `smore`. Hence, subsequent steps of the pipeline can be called several times with different parameters in order to have a comparison when using distinct similarity thresholds or levels of joining clusters. Calling `smore prep` is similar as calling `smore bake` except that the option for the phylogenetic tree is not needed within `smore prep`. A call for `smore prep` with `tRNAscan-SE` is written as

```
smore prep --out OUTPATH/Output_smoreprep --ref hg38 --trna
             --maf PATH_TO_MAF_FILES --genomes PATH_TO_GENOMEFOLDER
```

For further details on `smore prep` parameter, see Sec. .

6.4.1 .. `smore toast`

Given `smore prep` output, the subcommand `smore toast` can be started easily by referring to `smore prep` output with `--prep` option. Hence, a simple `smore toast` call will look like:

```
smore toast --out OUTPATH/Output_smoretoast --prep OUTPATH/Output_smoreprep
             --newick tree_6primates.newick
```

The `smore toast` has several additional optional parameter, that will change the outcome of the analysis, such as similarity thresholds and the level of joining clusters. Other options will increase the number of output files such that the user can have a closer look into the intermediary data. While more verbose output will increase running time of the program, other options such as skipping checks for deletions will decrease the running time. Please see Sec. for more details on parameters.

6.4.2 .. `smore mix, roast, eat`

This combination of subcommands is mostly for larger data sets or if the user wants to run several test runs with different combinations of parameters. Hereby, the user can check intermediary files

without running the whole pipeline and repeat steps directly. In case of a large data set, `smore mix` can split the data. This will be shown in the following toy example based on the tRNA data of 6 primate species as above.
Given the `smore prep` output, we now start `smore mix`. In order to reduce the running time of the next step, we want to have at most 400 clusters in each instance of execution. Hence, we call `smore mix` with the following options:

```
smore mix --out OUTPATH/Output_smoremix --prep OUTPATH/Output_smoreprep
          --newick tree_6primates.newick
          --max 400 --species OUTPATH/Output_smoreprep/specieslist
```

The parameter `--species` requires a file with species identifiers, one in each line. This file is created by the `smore prep` run and will be located in `smore prep` output files.

`smore mix` output include a file called `commandlist` and a subfolder called `clusterlists` which contains the complete clusterlist divided into several files. The commands in the commandlist can be copied and directly used to call the next step of the pipeline `smore roast`. For our example, the list of clusters was divided into three sublists, hence the commandlist lists the following commands:

```
(1)smore roast --tool TOOLPATH/SMORE --out OUTPATH/Output_smoreroast
                --python /usr/bin --perl /usr/bin -s 0.8 -p 0.8
                --in OUTPATH/Output_smoremix/clusterlists/clusList_part0.txt
                --newick tree6species.newick
                --species OUTPATH/Output_smoreprep/specieslist
(2)smore roast --tool TOOLPATH/SMORE --out OUTPATH/Output_smoreroast
                --python /usr/bin --perl /usr/bin -s 0.8 -p 0.8
                --in OUTPATH/Output_smoremix/clusterlists/clusList_part1.txt
                --newick tree6species.newick
                --species OUTPATH/Output_smoreprep/specieslist
(3)smore roast --tool TOOLPATH/SMORE --out OUTPATH/Output_smoreroast
                --python /usr/bin --perl /usr/bin -s 0.8 -p 0.8
                --in OUTPATH/Output_smoremix/clusterlists/clusList_part2.txt
                --newick tree6species.newick
```

As it can be seen, the calls are exactly the same except for the cluster list file. Even though the output folder is the same, output files will have unique names by including the name of the corresponding cluster list. It is important, that output files of all `smore roast` runs that belong to the same original cluster list are located in the same folder as the next step of the pipeline, `smore eat` will summarize all those files.

Hint: The output folders of `smore roast` and `smore mix` can be the same as the next step, `smore eat` will need the locations of the folders.

After running `smore roast` for all clusters lists, `smore eat` will summarize all output files and output the final results such as the phylogenetic tree with event counts.

```
(1)smore eat --out OUTPATH/Output_smoreeat
             --prep OUTPATH/Output_smoreprep
             --mix OUTPATH/Output_smoremix
             --roast OUTPATH/Output_smoreroast
             --newick tree6species.newick
             --species OUTPATH/Output_smoreprep/specieslist
```

As the example data is the same as in all other examples above, the output, e.g. the phylogenetic tree, will be the same as above.

7 Theory

This section will explain some theoretical parts that might be needed to understand the internal processes of `smore`. For further details and references, we refer to the publication [2].

7.1 Graph Representation

As explained in the corresponding publication [2], graph structures are used to represent orthology relations. Hence, nodes in the graphs are genetic elements. There is an edge between two nodes if the corresponding genes are orthologs (i.e. fulfill the similarity thresholds and genes are from different species, hence orthologs). I was shown that a valid orthology relation should have a cograph structure [6] i.e., it must not include a path P_4 on four vertices as an induced subgraph. As input data might contain errors or there are some outliers in the distribution of similarity thresholds, the cograph structure might not always be given. In order to obtain a valid orthology relation, non-cographs are corrected by editing as less as possible edges. Fig. 7 A shows an example.
Additionally, it is useful to know how graph structures look like on average. Hence, number of nodes and edges as well as densities are collected and an averaged value is contained in the output. The density of a graph usually acts as an indicator for the number of edges in comparison to the number of nodes. Given an orthology graph with only edges from the same species (paralogs), no edges will be drawn. Hence, even though genes are very similar, the density of the graph will be zero. Hence, for density calculations, we use a *corrected* number of edges, which only takes into account similarity thresholds and not the fact if genes are from the same species or not. This corrected edge number is included in cograph and non-cograph list to give a better intuition about the graph structure.

After correction graphs for cograph structure, clusters undergo the duplication alignment. Here, a sequence of genetic elements for each species in the cluster is created. Each genetic element is represented by one letter. If two elements fulfill the similarity thresholds, they are assigned the same letter, for all of the species and inside a species, too. The order of elements in a cluster sequence depends on the order of genes in the corresponding genome. Given two cluster sequences from different species, an alignment is created such that insertions and deletions can be detected.

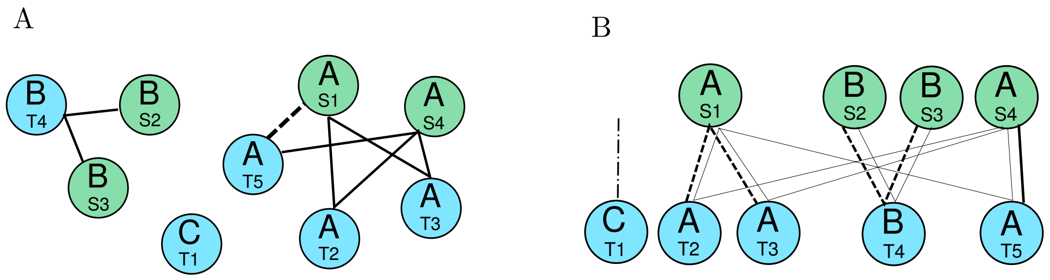


Figure 7: Example of the graph G for a cluster consisting of two groups of orthlogous elements in two species S and T (A). Thick edges indicate above-threshold sequence similarity. The dashed edge, which was included initially must be inserted to correct G: otherwise T5-S4-T3-S1 would form a P_4. Modified Needleman-Wunsch alignment for graph G (B). The inserted edge to correct for a cograph is now part of the thick edges showing the orthology relation. The alignment will remove crossing edges of the orthology graph and detect duplications (dashed edges). The edge attached to node T_1 indicates a deletion in species S as there is no target node for this edge.

Additionally, the modified alignment algorithm allows to align one element in the first species to more than one element in the other species if all element are of the same type and adjacent to each other when in the same species. This is shown in Fig. 7 B.

7.2 Counting

Given a set of genetic elements per species, they will be assigned to an evolutionary event in the phylogenetic tree. After collecting the information of all genetic elements, the assignments will be counted and the corresponding evolutionary event together with the number of elements will be inserted in the tree. The following evolutionary events can be found in the resulting tree (for a figure showing the events, we refer to the example section):

- *Total*: This shows the total number of elements that were given as an input or contained in the resulting gene list based on the covariance model.
- *Excluded*: these genetic elements could not be assigned in between two genetic anchors. Hence, they are excluded from the analysis. In order to fit the total number of elements per species, they are still included in the resulting tree.
- *Singleton*: These elements appear as the only elements in the defined cluster. Additionally, no orthologuous cluster was detected. Hence, it is not possible to set those elements in the evolutionary context, so they are counted for each species separately at the leaves of the tree.
- *Duplication*: Duplications are detected by applying the modified list alignments. Thus, they are in a cluster together with a second highly similar element (or copy) in the same species.

Other species might include only one copy of this element. Thus, the additional element is counted as a duplication.

- *Insertion*: Given orthologuous elements in several species, the element is assumed to have appeared in a common ancestor.
- *Deletion*: An element is said to be deleted if several related species have an orthologuous version of it but it cannot be found in the orthologuous cluster of the current species.
- *Missing Data*: In case of a deletion but the absence of orthologuous anchors in the current species, this is count as missing data as this could also be induced by low quality of the multiple sequence alignments or underlying sequences.
- *Pseudogene*: Pseudogenizations are not explicitly stated in the tree but within each evolutionary event, we give a number (in parentheses) that gives the number of pseudogenes counted at this node of the tree. Hence, pseudogenes are included in the analysis but counted separately.

8 Help Pages

The `smore` pipeline can be started using general options with

```
smore [general options]
```

or

```
smore <subcommand> [options]
```

For both cases, calling `--help` as option, the help pages will be displayed. Additionally, there is an extra help page explaining output files that can be called using `--helpout`.

9 Contact

When using `smore`, please cite

```
SMORE: Synteny Modulator Of Repetitive Elements.
Berkemer, S.J. and Hoffmann, A. and Murray, C.R. and Stadler, P.F.
MDPI Life, 2017 (submitted)
```

In case of further questions, please contact:

```
bsarah at bioinf dot uni-leipzig dot de
anneh at bioinf dot uni-leipzig dot de
```

References

[1] Velandia-Huerto, C.A., Berkemer, S.J., Hoffmann, A., Retzlaff, N., Romero Marroquín, L.C., Hernández Rosales, M., Stadler, P.F., Bermúdez-Santana, C.I.: Orthologs, turn-over, and remolding of tRNAs in primates and fruit flies. BMC Genomics **17**, 617 (2016). doi:10.1186/s12864-016-2927-4

[2] Berkemer, S.J., Hoffmann, A., Murray, C.R., Stadler, P.F.: SMORE: Synteny Modulator Of Repetitive Elements. MDPI Life (2017 submitted)

[3] Nawrocki, E.P., Eddy, S.R.: Infernal 1.1: 100-fold faster rna homology searches. Bioinformatics **29**, 2933–2935 (2013)

[4] Lowe, T.M., Eddy, S.R.: tRNAscan-SE: A program for improved detection of transfer RNA genes in genomic sequence. Nucleic Acids Res. **25**, 955–964 (1997). doi:10.1093/nar/25.5.0955

[5] Letunic, I., Bork, P.: Interactive tree of life (itol) v3: an online tool for the display and annotation of phylogenetic and other trees. Nucleic acids research **44**(W1), 242–245 (2016)

[6] Hellmuth, M., Hernandez-Rosales, M., Huber, K.T., Moulton, V., Stadler, P.F., Wieseke, N.: Orthology relations, symbolic ultrametrics, and cographs. J. Math. Biol. **66**, 399–420 (2013). doi:10.1007/s00285-012-0525-x