

Modules

absl

numpy

os

Functions

main()

Modules

RNA

numpy

os

Functions

canon_bp(i, j)

get_bp_counts(seqs, structs)

julia_looptypes(seqs, structs)

julia_prediction(seqs, data)

Wenn ich richtig verstehe, wird ab Zeile 36 eine 0,1 Matrix gebaut, die 1 Einträge enthält wenn der NN output >0.5 war und der Eintrag der größte auf der Zeile ist. Kling gut, ausser dass auch pro Spalte höchstens eine 1 stehen darf. Wird die NN Matrix vielleicht vorher schon symmetrisch gemacht?

Der code failed auch, wenn der Maximalwert in einer Zeile doppelt vorkommt, aber das ist hoffentlich selten genug.

julia_version(a)

main()

remove_conflicts(a, seq=None)

Modules

numpy

os

random

Functions

main()

Generates random sequence data files.

datadir/datatype.fasta

...

>rseq_{i}_{energy}

sequence

structure

>rseq_{i+1}_{energy}

sequence

structure

...

Modules

absl

argparse

os

Functions

```
main()
RNAdeep training interface.
python train.py -d l30 -t train_data/fixlen30_n100000.fa-train -v train_data/fixlen30_n100000.fa-valid -m 4 -b 250 -e 20 --model-
log-dir intermediate_models -l intermediate_models/sm4_l30_010/ --epoch0 10 2>> sm4_l30.err >> sm4_l30.out &

parse_rnadeep_args(p)
Arguments that are used by RNAdeep.

training(datatag, ftrain, fvalid, spotmodel=None, basemodel=None, savedir='.', epochs=50, epoch0=0, batch_size=4)
```

predict

[index](#)
[rnadeep/examples/predict.py](#)

Modules

[absl](#) [numpy](#) [os](#)

Functions

```
main()
```

train_ali (version v0.1)

[index](#)
[rnadeep/examples/train_ali.py](#)

Modules

[absl](#) [argparse](#) [os](#) [tensorflow](#)

Functions

```
main()
RNAdeep training interface.

parse_rnadeep_args(p)
Arguments that are used by RNAdeep.

training(datatag, dbn_dir, ali_dir, spotmodel=None, basemodel=None, savedir='.', epochs=50, epoch0=0, batch_size=4)
```

rfam_filter

[index](#)
[rnadeep/rnaconv/rfam_filter.py](#)

Modules

[os](#) [sys](#)

Functions

```
filter_rfam_data(ali_dirpath, single_freq_dirpath, doublet_freq_dirpath, neigh_wuss_dirpath, neigh_dbn_dirpath,
tree_fixed_dirpath, tree_rescaled_dirpath, max_length=700)
Filters the alignments, consensus structures, frequencies and trees of a converted rfam database (not touching
the original tree files and original Rfam.seed file):
Every data point (consisting of the four filetypes mentioned above) which alignment exceeds a certain length is
removed.

Parameters:
ali_dirpath (str): Path to the directory of the converted alignments.
single_freq_dirpath (str): Path to the directory of the extracted single frequency files.
doublet_freq_dirpath (str): Path to the directory of the extracted doublet frequency files.
neigh_wuss_dirpath (str): Path to the directory of the extracted wuss files.
neigh_dbn_dirpath (str): Path to the directory of the extracted dbn files.
tree_fixed_dirpath (str): Path to the directory of the fixed newick string tree files.
tree_rescaled_dirpath (str): Path to the directory of the rescaled tree files.
max_length (int, None, optional): Maximum allowed length of an alignment. Default is 700.

main()
```

data_filter

[index](#)
[rnadeep/rnaconv/data_filter.py](#)

Modules

[RNA](#) [numpy](#) [os](#) [sys](#)

Functions

```
filter_data(ali_dirpath, seq_dirpath, neigh_dbn_dirpath, neigh_ct_dirpath, max_dbars_deviation=20)
Filters the data generated by the data_generator: In each alignment, every sequence is removed that
```

deviates by over <max_dbrs_deviation> from the consensus structure that was used by SISSI to generate it. This is achieved by using RNAfold to predict the secondary structure and the base pair distance to compare it to the desired consensus structure. If this results in all sequences being removed, the whole alignment with the corresponding consensus structure and sequence is removed.

Note: If families were generated, the consensus structures used for the alignment generation were generated by RNAfold and saved into the neigh_dirpath.

If only alignments were generated, the consensus structures used for the generation were provided by the user, most likely from a converted Rfam database, but then copied to the neigh_dirpath anyway, for the sake of integrity.

Therefore, in both cases, neigh_dirpath can be used to retrieve the desired consensus structures to compare the sequences with.

Parameters:

ali_dirpath (str): Path to the directory of the generated alignments.
seq_dirpath (str): Path to the directory of the generated or copied sequences.
neigh_dbn_dirpath (str): Path to the directory of the generated or copied dbn files.
neigh_ct_dirpath (str): Path to the directory of the generated ct files.
max_dbrs_deviation (int, None, optional): maximum allowed base pair distance deviation from the consensus structure in percent. Default is 20.

main()

obtain_and_compare_equilibrium_frequencies(ali_dirpath, neigh_dirpath, orig_single_freq_dirpath, orig_doublet_freq_dirpath, outpath)

Extracts the equilibrium frequencies for unpaired single nucleotides and nucleotide pairs from the generated alignments and forms the differences to the already extracted equilibrium frequencies of the original alignments.

Parameters:

ali_dirpath (str): Path to the directory of the generated alignment files in CLUSTAL format
neigh_dirpath (str): Path to the directory containing the alignment consensus structure files in dot bracket notation format
orig_single_freq_dirpath (str): Path to the directory of the extracted single frequency files of the original alignments
orig_doublet_freq_dirpath (str): Path to the directory of the extracted doublet frequency files of the original alignments
outpath (str): Path to the directory in which to save the extracted unpaired single and paired nucleotide equilibrium frequencies and frequency differences

data_generator

[index](#)
rnadeep/rnaconv/data_generator.py

Modules

[RNA](#)
[argparse](#)

[os](#)
[random](#)

[subprocess](#)

Functions

db_to_ct(dbn, seq)

Converts the consensus structures contained in the dot bracket notation input file into the connect table format.

Parameters:

dbn (str): Secondary structure in dot bracket notation
seq (str): Sequence

generate_alignment_set(sissi_filepath, number, tree_dirpath, neigh_dirpath, sfreq_dirpath, dfreq_dirpath, ali_dirpath, outpath)

Generates <number> alternative alignments for each tree file in the given tree-directory, searching in the respectively given consensus-structure-, single- & doublet-frequencies- and, optionally for reading ndels, alignment-directories for files of the same name to use.

Parameters:

sissi_filepath (str): Path to the compiled sissi099 file
number (int): The number of alignments to generate
tree_dirpath (str): Path to a directory containing tree files in the newick string format ('.seed_tree')
neigh_dirpath (str): Path to a directory containing neighbourhood files in the dot-bracket notation format ('.dbn')
sfreq_dirpath (str): Path to a directory containing files storing a single frequency vector ('.sfreq')
dfreq_dirpath (str): Path to a directory containing files storing a doublet frequency vector ('.dfreq')
ali_dirpath (str, None): Path to a directory containing alignment files in the clustal format ('.aln')
outpath (str): The Path to which to write the generated sequences, alignments & copied consensus structures

generate_alignments(sissi_filepath, number, tree_filepath, neigh_filepath, sfreq_filepath, dfreq_filepath, ali_filepath, outpath)

Generates <number> alternative alignments for the given tree-, consensus-structure-, single- & doublet-frequencies- and, optionally for reading indels, alignment-file, using:

- RNAinverse to generate an ancestral sequence for the provided consensus structure
- SISSI simulate homologous sequence alignments (taking the generated ancestral sequence, provided tree, provided consensus structure and provided equilibrium frequencies as input).

Note:

The provided consensus structure will also be copied into an additional file per generated alignment, in order to create pairs of samples and tags to be used for training (during the process, the dbn files are converted to ct files, which are also saved).

The generated ancestral sequence will also be saved to maintain integrity.

Parameters:

sissi_filepath (str): Path to the compiled sissi099 file
number (int): The number of alignments to generate
tree_filepath (str): Path to a directory containing tree files in the newick string format ('.seed_tree')
neigh_filepath (str): Path to a directory containing neighbourhood files in the dot-bracket notation format ('.dbn')
sfreq_filepath (str): Path to a directory containing files storing a single frequency vector ('.sfreq')
dfreq_filepath (str): Path to a directory containing files storing a doublet frequency vector ('.dfreq')
ali_filepath (str, None): Path to a directory containing alignment files in the clustal format ('.aln')
outpath (str): The Path to which to write the generated sequences, alignments & copied consensus structures

generate_families(sissi_filepath, number, min_length, max_length, tree_filepath, sfreq_filepath, dfreq_filepath, outpath)

Generates <number> families for the given tree- and single- & doublet-frequencies-file, using:

- random ancestral sequences of uniformly distributed lengths up to <maxlength>
- RNAfold to predict secondary structures for these sequences to be used as consensus structures for the alignment generation
- SISSI simulate corresponding homologous sequence alignments (taking the random ancestral sequences, provided tree, predicted consensus structures, and provided equilibrium frequencies as input).

Note:

During the process, the generated dbn files are converted to ct files, which are also saved.

Parameters:

sissi_filepath (str): Path to the compiled sissi099 file
number (int): The number of families to generate
min_length (int): Minimum allowed length of the ancestral sequences used to generate the families
max_length (int): Maximum allowed length of the ancestral sequences used to generate the families
tree_filepath (str): Path to a tree file in the newick string format ('.seed_tree')
sfreq_filepath (str): Path to a file containing a single frequency vector ('.sfreq')
dfreq_filepath (str): Path to a file containing a doublet frequency vector ('.dfreq')
outpath (str): The path to which to write the generated families

generate_family_set(sissi_filepath, number, min_length, max_length, tree_dirpath, sfreq_dirpath, dfreq_dirpath, outpath)

Generates <number> RNA families of uniformly distributed lengths up to <maxlength> for each tree file in the given tree-directory, searching in the respectively given single- & doublet-frequencies-directories for files of the same name to use.

For more information, refer to the [generate_families\(\)](#) function.

Parameters:

sissi_filepath (str): Path to the compiled sissi099 file
number (int): The number of families to generate
min_length (int): Minimum allowed length of the ancestral sequences used to generate the families
max_length (int): Maximum allowed length of the ancestral sequences used to generate the families
tree_dirpath (str): Path to a directory containing tree files in the newick string format ('.seed_tree')
sfreq_dirpath (str): Path to a directory containing files storing a single frequency vector ('.sfreq')
dfreq_dirpath (str): Path to a directory containing files storing a doublet frequency vector ('.dfreq')
outpath (str): Path to which to write the generated families

generate_sequence_structure_pair(length=85, min_paired_sites_percent=20)

Repeatedly generates a random sequence and predicts its secondary structure using RNAfold, until the structure has at least min_paired_sites paired sites.

Parameters:

length (int, optional): Length of the random sequence
min_paired_sites_percent (int, optional): Minimal required sites to be paired in percent

main()

setup_args(parser)

rfam_converter

[index](#)
[rnadeep/rnaconv/rfam_converter.py](#)

Modules

[RNA](#)
[numpy](#)

[os](#)
[shutil](#)

[sys](#)
[textdistance](#)

Functions

convert_rfam_data(seed_filepath, tree_dirpath, outpath)

Calls the necessary functions to convert the whole rfam database into single files, preparing them to be used by SISSI.

Parameters:

seed_filepath (str): Path to the Rfam.seed file in STOCKHOLM format, containing the families
tree_dirpath (str): Path to the directory in which Rfam tree files in newick format (.seed_tree) files are located
outpath (str): Path to the directory in which to save the converted data

fix_newick_strings(tree_dirpath, outpath)

Fixes newick strings by replacing every control character (e.g. '(', ')', ',', '.', ':') within a node name with an underscore.

Additionally, multifurcations are resolved and non-leaf node labels are removed.
(These three steps are nessecary for SISSI to be able to parse the Rfam tree files.)

Parameters:

tree_dirpath (str): path to the directory containing the tree files in newick string format
outpath (str): path to the directory in which to save the trees in the fixed newick string format.

main()

obtain_equilibrium_frequencies(ali_dirpath, neigh_dirpath, outpath)

Extracts the equilibrium frequencies for unpaired single nucleotides and nucleotide pairs from an alignment, by counting the occurences of single nucleotides in unpaired site and saving them in a 4-vector, counting the occurences of nucleotide pairs in paired sites and saving them in a 16-vector, adding pseudocounts to both (+1 for each element) and normalizing in the end.

Parameters:

ali_dirpath (str): Path to the directory containing the alignment files in CLUSTAL format
neigh_dirpath (str): Path to the directory containing the alignment consensus structure files in dot bracket notation format
outpath (str): Path to the directory in which to save the extracted unpaired single and paired nucleotide

equilibrium frequencies

rescale_newick_strings(tree_dirpath, ali_dirpath, outpath)

Rescales the tree branch lengths for trees which corresponding sequence alignments sequences are over 95% similar with respect to their mean pairwise hamming distance, in order to increase the evolution rate when using the tree for evolutionary simulation.

The rescale factor is 2.

Parameters:

tree_dirpath (str): path to the directory containing the tree files in newick string format

ali_dirpath (str): path to the directory containing the alignment files in CLUSTAL format

outpath (str): path to the directory in which to save the rescaled trees in the newick string format.

stockholm_to_alignments(filepath, outpath)

Converts the alignments contained in the STOCKHOLM input file into CLUSTAL files.

Parameters:

filepath (str): Path to the Rfam.seed file in STOCKHOLM format, containing the families

outpath (str): Path to the directory in which to save the extracted alignments in the CLUSTAL format

stockholm_to_neighbourhoods(filepath, outpath)

Calls the necessary functions to convert the consensus structures contained in the STOCKHOLM input file into single files in the wuss and dbn formats, respectively.

Parameters:

filepath (str): Path to the Rfam.seed file in STOCKHOLM format, containing the families

outpath (str): Path to the directory in which to save the extracted consensus structures

stockholm_to_wuss(filepath, outpath)

Converts the consensus structures contained in the STOCKHOLM input file into single files in the washington university secondary structure (wuss) format.

Parameters:

filepath (str): Path to the Rfam.seed file in STOCKHOLM format, containing the families

outpath (str): Path to the directory in which to save the resulting wuss file

wuss_to_db(filepath, outpath)

Converts the consensus structures contained in the wuss input file into the dot bracket notation format

Parameters:

filepath (str): Path to the file in wuss format, containing the secondary structure

outpath (str): Path to the directory in which to save the resulting dot bracket notation file

lstm_models

[index](#)

[rnadeep/rnadeep/lstm_models.py](#)

Modules

[keras.api.v2.keras.backend](#)

[numpy](#)

[os](#)

[tensorflow](#)

Functions

blstm(lstm_layers=1, lstm_neurons=20)

complex_blstm(lstm_layers=1, lstm_neurons=40)

metrics

[index](#)

[rnadeep/rnadeep/metrics.py](#)

Modules

[keras.api.v2.keras.backend](#)

[numpy](#)

[tensorflow](#)

Functions

f1(y_true, y_pred)

focal_loss(gamma=2.0, alpha=0.75)

matthewscorrelation(y_true, y_pred)

mcc(y_true, y_pred)

sensitivity(y_true, y_pred)

specificity(y_true, y_pred)

__init__ (version v0.1)

[index](#)

[rnadeep/rnadeep/_init_.py](#)

sliding_window

[index](#)

[rnadeep/rnadeep/sliding_window.py](#)

Modules

[keras.api.v2.keras.backend](#) [numpy](#) [tensorflow](#)

Functions

basic_window(window_size)

basic_window_leakyrelu(window_size)

conv_window(window_size)

Data

absolute_import = _Feature((2, 5, 0, 'alpha', 1), (3, 0, 0, 'alpha', 0), 262144)
division = _Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 131072)
print_function = _Feature((2, 6, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 1048576)
unicode_literals = _Feature((2, 6, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 2097152)

models

[index](#)
[rnadeep/rnadeep/models.py](#)

Modules

[keras.api.v2.keras.layers](#) [tensorflow](#)

Functions

spotrna_alignment_models(model=1, use_mask=True)
Some modifications to Julia's SPOT-RNA implementations.

Supposed to be a reimplementaion of the models in the SPOT-RNA paper. If you find mistakes, please let us know!

Overview:

- Initial 3x3 convolution layer
- ResNet blocks
- Act./Norm.
- 2D-BLSTM
- Fully Connected blocks
- Output masking layer (optional)
- Output layer

Args:

model: select the model (0-4)
use_mask: for padded input/output (defaults to True!)

spotrna_models(model=1, use_mask=True)
Some modifications to Julia's SPOT-RNA implementations.

Supposed to be a reimplementaion of the models in the SPOT-RNA paper. If you find mistakes, please let us know!

Overview:

- Initial 3x3 convolution layer
- ResNet blocks
- Act./Norm.
- 2D-BLSTM
- Fully Connected blocks
- Output masking layer (optional)
- Output layer

Args:

model: select the model (0-4)
use_mask: for padded input/output (defaults to True!)

encoding_utils

[index](#)
[rnadeep/rnadeep/encoding_utils.py](#)

Modules

[numpy](#)

Functions

base_pair_matrix(ss)

binary_encode(structure)

create_windows(sequences, window_size)

encode_padded_alignment_matrix(alignments, max_length=None)

encode_padded_sequence_matrix(sequences, max_length=None)

encode_padded_structure_matrix(structures, max_length=None)

encode_sequence(sequences)

encode_sequence_matrix(sequences)

Make a BP probability matrix with one-hot encoding of basepairs.
NOTE: This only works if all sequences have the same length, otherwise
you need to use: encode_padded_sequence_matrix

encode_sequence_windows(sequences, window_size)

encode_structure(structures)

encode_structure_matrix(structures)

Make a BP probability matrix with one-hot encoding of basepairs.
NOTE: This only works if all sequences have the same length!

make_pair_table(ss, base=0, chars=['.'])

Return a secondary struture in form of pair table.

Args:

ss (str): secondary structure in dot-bracket format
base (int, optional): choose between a pair-table with base 0 or 1
chars (list, optional): a list of characters to be are ignored, default:
['.']

Example:

base=0: ((..)). => [5,4,-1,-1,1,0,-1]
i.e. start counting from 0, unpaired = -1
base=1: ((..)). => [7,6,5,0,0,2,1,0]
i.e. start counting from 1, unpaired = 0, pt[0]=len(ss)

Returns:

[list]: A pair-table

one_hot_encode(char)

one_hot_matrix(seq)

profile_vec_matrix(ali)

Creates a profile matrix for the given alignment: For each cell a_ij, the columns i and j or the alignment are combined by forming the outer product of two profile vectors for the two respective symbols at the current row index of the two columns and summing them all up. The two respective profile vectors created by the sheme defined in the base_to_ids dictionary variable.

sampling_ali

[index](#)
[rnadeep/rnadeep/sampling_ali.py](#)

Modules

[numpy](#) [os](#)

Functions

draw_ali_sets(ali_directory, dbn_directory, splits=None)

parse_families(ali_dirpath, dbn_dirpath)

Combines pairs of the same name of alignment CLUSTAL files and neighbourhood Dot Bracket String files found in the respective directories to be used for training.

Parameters:

ali_dirpath (str): Path to the directory containing the alignment CLUSTAL files
dbn_dirpath (str): Path to the directory containing the neighbourhood Dot Bracket String files

parse_family(ali_filepath, dbn_filepath)

Reads an alignment CLUSTAL file and neighbourhood Dot Bracket String file and combines them into a pair to be used for training.

Parameters:

ali_filepath (str): Path to the alignment CLUSTAL file
dbn_filepath (str): Path to the neighbourhood Dot Bracket String file

sampling

[index](#)
[rnadeep/rnadeep/sampling.py](#)

Modules

[numpy](#) [os](#) [random](#)

Functions

draw_sets(fname, splits=None)

generate_random_structures(lengths)

rseq(l)

write_data_file(data, fname, mode='w')

Save sequence/structure pairs for the given lengths.

write_fixed_len_data_file(seqlen, num, root='')

write_normal_len_data_file(central, std, num, root='')

write_uniform_len_data_file(minlen, maxlen, num, root='')