predict_ali

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/examples/predict_ali.py

Modules

absl	numpy	os

Functions

main()

mlforensics

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/examples/mlforensics.py

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)

generate_data

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/examples/generate_data.py

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

'''
```

train (version v0.1)

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/examples/train.py

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
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/examples/predict.py

Modules

absl numpy os

Functions

main()

train_ali (version v0.1)

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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)

alignment_filter

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/alignment_filter.py

Modules

RNA numpy os sys

Functions

filter_alignments(path, rfam_path, max_dbrs_deviation)

main()

obtain_sissi_frequencies(path, rfam_path)

Data

default_max_dbrs_deviation = 20

rfam_filter

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/rfam_filter.py

Modules

os sys

Functions

filter_rfam_data(rfam_path, max_length)

main()

Data

default_max_length = 700

family_filter

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/family_filter.py

Modules

RNA numpy os sys

Functions

filter_alignments(path, max_dbrs_deviation)

main()

Data

default_max_dbrs_deviation = 20

alignment_generator

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/alignment_generator.py

Modules

RNA os subprocess sys

Functions

db_to_ct(dbn, seq)

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

generate_alignments(sissi_filepath, n, tree_filepath, neigh_filepath, sfreq_dfilepath, dfreq_filepath, ali_filepath, outpath)

```
Generates n RNA alignments using sissi for given equilibrium frequencies, neighbourhood system and phylogenetic tree.
The raw alignments are used to re-add indels.
Note: The given same neighbourhood will also be copied once for each generated alignment to create pairs for
easier parsing into the network.

Parameters:
sissi_filepath (str): path to the compiled sissi099 file
n (int): The number of alignments to generate
tree_filepath (str): path to a tree file in the newick string format ('.seed_tree')
neigh_filepath (str): path to a neighbourhood file in the sissi01 format ('.nei')
sfreq_dfilepath (str): path to a file containing a single frequency vector ('.sfreq')
dfreq_filepath (str): path to a file containing a doublet frequency vector ('.dfreq')
ali_filepath (str): path to a file containing an alignment in the clustal format ('.aln')
outpath (str): The path to which to write the generated alignments
```

get_paths(rfam_path)

main()

rfam_converter

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/rfam_converter.py

Modules

RNA	os	subprocess	textdistance
numpy	shutil	sys	

Functions

convert_rfam_data(seed_filepath, ali_outpath, neigh_outpath, freq_outpath, tree_path, tree_fixed_outpath, tree_rescaled_outpath)

ct_to_nei(filepath, outpath)

db_to_ct(filepath, outpath)

fix_newick_strings(treedirpath, outpath)

main()

obtain_equilibrium_frequencies(alidirpath, neighdirpath, outpath)

rescale_newick_strings(treedirpath, alidirpath, outpath)

stockholm_to_alignments(filepath, outpath)

stockholm_to_neighbourhoods(filepath, outpath)

stockholm_to_wuss(filepath, outpath)

wuss_to_db(filepath, outpath)

family_generator

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnaconv/family_generator.py

Modules

RNA	random	sys
os	subprocess	

Functions

db_to_ct(dbn, seq)

generate_family(sissi_filepath, n, length, tree_filepath, sfreq_filepath, dfreq_filepath, outpath)

```
Generates n RNA families (consisting of an alignment and a secondary structure)
for given equilibrium frequencies and a phylogenetic tree, using:
- a random ancestral sequence
- RNAfold to predict a consensus structure
- SISSI simulate a corresponding homologous sequence alignment.

Parameters:
sissi_filepath (str): path to the compiled sissi099 file
n (int): The number of families to generate
length(int): Length of the ancestral sequence used to generate the family
tree_filepath (str): path to a tree file in the newick string format ('.seed_tree')
sfreq_dfilepath (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, n, length, tree_dirpath, sfreq_dirpath, dfreq_dirpath, outpath)

generate_sequence_structure_pair(length=85, min_paired_sites=0)

get_paths(rfam_path)

main()

Data

default_min_paired_sites = 25

lstm_models

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnadeep/__init__.py

sliding_window

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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 reimplementation 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 reimplementation 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
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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)

sampling_ali

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/rnadeep/rnadeep/sampling_ali.py

Modules

numpy os

Functions

draw_ali_sets(ali_directory, dbn_directory, splits=None)

parse_alignment(ali_path, dbn_path, filename)

parse_alignments(ali_directory, dbn_directory)

sampling

index
/home/julian-zim/Files/Cloud/OneDrive/OneFiles/Linux/Work/Workspaces/Study/UNIVIE/PyCharm/PR_SPB/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='')