Overview of NADDA

NADDA(Abnousi, Broschat, and Kalyanaraman 2016) uses k-mer frequeuncies of the protein sequences in an input dataset to generate a vectorized representation of each protein. Then it utilizes the generated representative vectors, called k-mer ferquency profiles, as features for prediction of conserved regions, i.e., domains and motifs in the sequences. This package (*nadda*) provides the functions required for generation of instances that can be used to train a new model for conserved index prediction and for generation of test instances that can be used as input to a trained model to predict the conserved indices on them.

Generating Training and Test Instances

To generate a training or test set, use the function *generate_instances()*. The *labeled* parameter for this function decides is to indicate whether a training set should be generated or a test set. For generation of a training set, the current version of the package uses Pfam or InterPro annotations. These annotations should be provided as a tsv (tabulated) file using the *truth_filename* parameter.

In the following example we generate a dataset of three protein sequences, calling them "seq1", "seq2", and "seq3". Then we use the *generate_instances* method to create a test set.

```
library(nadda)
```

```
## Generate a set of three example protein sequences
seqs <- AAStringSet(c("seq1"="MLVVD",
                      "seq2"="PVVRA",
                      "seq3"="LVVR"))
## Generate a test set using the dataset seqs
ins <- generate_instances(seqs, labeled = FALSE, parallel = FALSE, klen = 3, impute = TRUE, winlen = 5,
#> Warning in count_kmers.AAStringSet(obj, klen, parallel, nproc, distributed): number of processors, 1
#>                       count_kmers function. Using all available processors.
head(ins)
#>    name position freqn2 freqn1 freqindex freqp1 freqp2
#> 1: seq1        1    1.5    1.5       1.0    2.0    1.0
#> 2: seq1        2    1.5    1.0       2.0    1.0    1.5
#> 3: seq1        3    1.0    2.0       1.0    1.5    1.5
#> 4: seq1        4    2.0    1.0       1.5    1.5    1.5
#> 5: seq1        5    1.0    1.5       1.5    1.5    1.5
#> 6: seq2        1    1.5    1.5       1.0    2.0    1.0

# To generate a training set using Pfam annotation located at user/seq_pfam.tsv the following command c
# ins <- generate_instances(seqs, labeled = FALSE, parallel = FALSE, groundtruth = "Pfam",
#                               truth_filename = "user/seq_pfam.tsv",  klen = 3, impute = TRUE,
#                               winlen = 5, normalize = FALSE)
```

Note that when the functions are run in parallel, each processor will hold only a chunk of the data. In this case, to perform the training on the complete set of data, one will need to aggregate all chunks and input them to a training method. For this purpose we suggest writing all instances on a single file on the shared memory, using **pbdMPI::comm.write** method from the *pbdMPI* package. After writing all instances to file, then one can use a single processor to read the data and perform the training using a package of choice, either in parallel or in serial.

Other functionalities

In addition, one can use more low-level functions to generate k-mer frequency profiles or to count the number of kmers (create a histogram of the kmers in the dataset). For more information on how to utilize these functionalities, please refer to the documentation of the *generate_profiles* and *count_kmers* methods.

When the *count_kmers* method is run in parallel (by setting *parallel* or *distributed* logical parameters), it computes the local k-mer counts for a chunk of data available to each processor. Then *allgather* method from *pbdMPI* is used to communicate these frequencies between different processors. Finally each processor sums up all acquired frequencies, thus generating a dataframe that is equal in between all processors and holds the number of times each k-mer appears in the dataset (Note: if a k-mer appears more than once in one sequence, it is counted only once.)

References:

Abnousi, Armen, Shira L Broschat, and Ananth Kalyanaraman. 2016. "A Fast Alignment-Free Approach for de Novo Detection of Protein Conserved Regions." *PloS One* 11 (8). Public Library of Science: e0161338.