1 Supplementary materials

1.1 BoaG queries

Following are the BoaG queries that used in this work to retrieve annotations from protein sequences and clusters.

```
s: Sequence = input;
count: output collection[string] of string;

getTaxList := function(seq: Sequence) : string {
taxids := "";
foreach (i: int; s.annotation[i].tax_name != "")
taxids = taxids + s.annotation[i].tax_id + " ";
return taxids;
};

count[s.seqid] <<  getTaxList(s);
```

Figure 1: Generate list of all taxids for the entire NR database. The output of BoaG script used to generate the tree of life in Figure 3 in the main manuscript. The BoaG query and its result is publicly available on the cluster here: http://boa.cs.iastate.edu/boag/?q=boa/job/public/80

```
s: Sequence = input;
counts: output sum[int] of int;
foreach(i:int; def(s.cluster[i])){
if (s.cluster[i].similarity==95)
  counts [s.cluster[i].length] <<  1;}
```

Figure 2: Frequencies of proteins for different size at 95% similarity. Sequence is a top level type in BoaG language. Variable counts is an output aggregator that sums number of proteins in different size, i.e. number of amino acids. The BoaG query takes only two minutes and can be accessed here: http://boa.cs.iastate.edu/boag/?q=boa/job/public/96

```
s: Sequence = input;
protOut : output sum [string][string] of int;
distinctTax := function (seq: Sequence): int{
taxSet : set of string;
foreach(i:int; def(seq.annotation[i]))
add(taxSet,seq.annotation[i].tax_id);
return(len(taxSet));};
if (distinctTax(s) > 10)
  foreach(i:int; def(s.annotation[i]))
    protOut [s.annotation[i].defline][s.seqid]<<1;
```

Figure 3: BoaG script for top protein functions for proteins that have more than 10 distinct taxonomic assignments. The output of this query is used to generate the top protein functions. The BoaG query and its result is publicly available on the cluster here: http://boa.cs.iastate.edu/boag/?q=boa/job/public/40

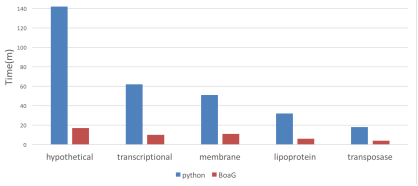
1.2 Programming efficiency in BoaG

BoaG programming performance in terms of lines of code and running time is shown in Figure 4.

BoaG

```
s: Sequence = input;
tax_stat : output count [string][
    string] of int;
foreach(i:int; def(s.annotation[i]))
  tax_stat[s.seqid] [s.annotation[i].
    tax_name]<<1;
```

Performance



Python

```
... // imports
tax_stat={}
with open(sys.argv[1], "r") as nr:
    for line in nr:
      deflines = line.split("\
  textbackslash\{\}x01")
      seq_id = line[1:line.index("
    ")]
      ... // get taxa frequency
  for each protein
    data=line.split(':')
... // iterate over all proteins
  in the NR database
files_list=[f for f in os.listdir(
  ".") if f.endswith('.txt')]
...
//... iterate over all proteins
sorted_d = sorted(tax_stat.items()
  , key=operator.itemgetter(1))
// ... return the top (n) taxa for
    each protein
```

Figure 4: Comparison of Line of Code(LOC) and performance to answer query "frequency of taxonomic assignment for each protein sequence" run on the different subset of the NR dataset. On the right side, the equivalent *Boa* code needs 33 lines of code in Python whereas the BoaG script needs only four.