`clustDist`, v. 0.8: Cluster Distances Using UPGMA or Neighbor-Joining

Bernhard Haubold
Max-Planck-Institute for Evolutionary Biology, Plön, Germany

November 6, 2018

1 Introduction

2 Getting Started

`clustDist` was written in C on a computer running Linux and should work on any standard UNIX system. However, please contact me at `haubold@evolbio.mpg.de` if you have any problems with the program.

- Unpack the program

  ```
  tar -xvzf clustDist_XXX.tgz
  ```

  where `XXX` indicates the version.
- Change into the newly created directory

  ```
  cd ClustDist_XXX
  ```

  and list its contents

  ```
  ls
  ```

- Generate `clustDist`

  ```
  make
  ```

- List its options

  ```
  ./clustDist -h
  ```

- Test program

  ```
  ./clustDist ecoli.dist
  ```

3 Listing

The following listing documents the driver program for `clustDist`.

```
/***** clustDist.c ********************************
 * Description: Cluster distances using UPGMA or NJ.
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Fri Dec 12 12:35:22 2014
 *************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <float.h>
#include <assert.h>
#include "gsl_rng.h"
#include "interface.h"
#include "eprintf.h"
#include "stringUtil.h"
#include "distTree.h"

void scanFile(FILE *fp, Args *args);
void cluster(Args *args, Matrix *mat, Node **tree);
void printNewickTree(Node *node);

int main(int argc, char *argv[]){
  int i;
  char *version;
  Args *args;
  FILE *fp;

  version = "0.8";
  setprogname2("clustDist");
  args = getArgs(argc, argv);
  if(args->v)
    printSplash(version);
  if(args->h || args->e)
    printUsage(version);
  if(args->numInputFiles == 0){
    fp = stdin;
    scanFile(fp, args);
  }else{
    for(i=0;i<args->numInputFiles;i++){
      fp = efopen(args->inputFiles[i],"r");
      scanFile(fp, args);
      fclose(fp);
    }
  }
  free(args);
  free(progname());
  return 0;
}

void permute(Node **tree, Matrix *mat, gsl_rng *r){
  int *index, i, n, j, x, y, tmp, rand;
  Node *tmpNode;

  n = mat->n;
  /* begin  debugging */
  printf("BEFORE␣Permute\n");
  for(i=0;i<n;i++){
    printf("%s",tree[i]->label);
    for(j=0;j<n;j++)
      printf("␣%.0f",mat->d[i][j]);
    printf("\n");
  }
  /* end debugging */
  index = (int *)emalloc(n*sizeof(int));
  for(i=0;i<n;i++)
    index[i] = i;
  for(i=n-1;i>=0;i--){
    rand = gsl_rng_uniform(r)*(i+1);
    tmp = index[i];
    index[i] = index[rand];
    index[rand] = tmp;
    tmpNode = tree[i];
    tree[i] = tree[rand];
    tree[rand] = tmpNode;
  }
  /* begin debugging */
  printf("Indexes:");
  for(i=0;i<n;i++)
    printf("␣%d",index[i]);
  printf("\n");
  /* end debugging */
  for(i=0;i<n-1;i++){
    for(j=i+1;j<n;j++){
      x = index[i];
      y = index[j];
      mat->d[j][i] = mat->d[x][y];
    }
  }
  for(i=0;i<n-1;i++)
    for(j=i+1;j<n;j++)
      mat->d[i][j] = mat->d[j][i];
  /* begin debugging */
  printf("AFTER␣Permute\n");
  for(i=0;i<n;i++){
    printf("%s",tree[i]->label);
    for(j=0;j<n;j++)
      printf("␣%.0f",mat->d[i][j]);
    printf("\n");
  }
  /* end debugging */
  free(index);
}


void scanFile(FILE *fp, Args *args){
  int i, j, n, c;
  Matrix *mat;
  Node **tree, *root;
  char *buf;
  gsl_rng *rng;

  buf = (char *)emalloc(1000*sizeof(char));
  c = 0;
  while(fscanf(fp,"%d",&n) != EOF){
    mat = newMatrix(n);
    tree = newTree(n);
    /* read distances */
    for(i=0;i<n;i++){
      assert(fscanf(fp,"%s",buf));
      tree[i]->label = strdup2(buf);
      mat->label[i] = strdup2(buf);
      for(j=0;j<n;j++)
        assert(fscanf(fp,"%lf",&(mat->d[i][j])));
    }
    /* average distances */
    for(i=1;i<n;i++)
      for(j=0;j<i;j++){
        mat->d[i][j] = (mat->d[i][j] + mat->d[j][i])/2.;
        mat->d[j][i] = mat->d[i][j];
      }
    if(c){
      if(c == 1)
        rng = ini_gsl_rng(args);
      /* permute(tree, mat, rng); */
    }
    cluster(args, mat, tree);
    if(args->u)
      root = tree[2*n-2];
    else
      root = tree[2*n-3];
    printNewickTree(root);
    freeMatrix(mat);
    freeTree(tree, n);
    c++;
  }
  if(c > 1)
    free_gsl_rng(rng,args);
  free(buf);
}
```

4 Change Log

- Version 0.1
  - First running version.
- Version 0.2 (December 6, 2012)
  - Compute average distances in case input matrix not symmetrical.
- Version 0.3 (February 6, 2013)
  - Allocated 512 instead of 256 bytes in

    `distTree.c: mat->label[t] = (char *)emalloc(512*sizeof(char));`

  This removed a segmentation fault. However, `valgrind.sh` shows that there seem to be problems with the initialization of variables.
  - Possibly related problem:

    ```
    ./cluster ../../Data/ecoli.dis | new2view
    error: syntax error
    ```

- Version 0.4 (December 12, 2014)
  - The syntax error above is gone (I have worked on `cluster` and `new2view` and am not quite sure how the error finally vanished).
  - Fixed the initialization problem flagged by `valgrind`.
  - Corrected tree printing.
  - Renamed program to `clustDist`.
  - Changed dangerous `strcat` to `strncat` in `distTree.constrTree`.
- Version 0.5 (January 5, 2015)
  - Fixed error in tree computation when applied to distance matrix of 2010 *E. coli* genomes by ensuring that no negative distances are computed; see line 220 in `distTree.c` (function `recalcDist`).
  - Fixed a memory error due to `strncat` in line 156 of `distTree.c`.
- Version 0.6 (June 11, 2015)
  - Allowed looping over multiple distance matrices.
  - In `phylip` the order of taxa is jumbled if more than one distance matrix is analyzed. I have tried to implement this in the function `permute`, but this clashes with `cluster` in some way that I don't understand yet. So for the time being jumbling is switched off.
- Version 0.7 (June 24, 2015)
  - Fixed deallocation of random number generator, which was a leftover from my failed attempt in version 0.6 to implement jumbling of taxa for multiple distance files.
- Version 0.8 (November 6, 2018)
  - Fixed bug in `interface.c`.