

# drawGenealogy, v. 1.4: Draw Genealogy of Diploid Individuals

Bernhard Haubold

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

November 6, 2018

## 1 Introduction

We all have two parents, four grand parents, eight great-grandparents, and so on. In fact, the theoretical number of our ancestors quickly exceeds the the number of people alive. As a result, in populations of size  $n$ , ancestors of all people alive today quickly appear in  $\approx \log_2(n)$  generations ?. Apart from such universal ancestors, there are individuals without any descendants in the present, and individuals who left at least one descendant in the present, but fewer than  $n$ . Intriguingly, such “partial ancestors” go extinct after  $\approx 1.77 \log_2(n)$  generations, leaving only universal ancestors and universal no-ancestors ??.

## 2 Getting Started

drawGenealogy 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](mailto:haubold@evolbio.mpg.de) if you have any problems with the program.

- Unpack the program

```
tar -xvzf drawGenealogy_XXX.tgz
```

where XXX indicates the version.

- Change into the newly created directory

```
cd DrawGenealogy_XXX
```

and list its contents

```
ls
```

- Generate drawGenealogy

```
make
```

- List its options

```
./drawGenealogy -h
```

### 3 Listing

The following listing documents the driver program for drawGenealogy.

```
1  /***** drawGenealogy.c *****/
   * Description:
   * Author: Bernhard Haubold, haubold@evolbio.mpg.de
   * File created on Fri Sep 23 09:37:15 2005.
   *****/
6  #include <stdio.h>
   #include <time.h>
   #include <stdlib.h>
   #include <string.h>
   #include <gsl/gsl_rng.h>
11  #include <gsl/gsl_randist.h>
   #include "gsl_rng.h"
   #include "interface.h"
   #include "eprintf.h"
   #include "stringUtil.h"
16  #include "drawGenealogy.h"

int main(int argc, char *argv[]){
    Args *args;          /* arguments */
    char *version;       /* program version */
21  int i, j, k, count;
    int tmrca;           /* time to the most recent common ancestor */
    int tia;             /* time to identical ancestors */
    gsl_rng *ranf;
    Indiv ***pop;
26  FILE *fp;
    int pick1, pick2, empty, full;

    version = "1.4";
    setprogname2("drawGenealogy");
31  args = getArgs(argc, argv);
    if(args->v)
        printSplash(version);
    if(args->h || args->e)
        printUsage(version);
36  /* setup random number generator */
    ranf = ini_gsl_rng(args);
    /* generate population making sure that there is at least one male & one
       female*/
    pop = (Indiv ***)emalloc(args->g*sizeof(Indiv **));
    for(i=args->g-1;i>=0;i--){
41  pop[i] = (Indiv **)emalloc(args->p*sizeof(Indiv *));
        count = 0;
        for(j=0;j<args->p;j++){
            pop[i][j] = getIndiv(ranf,args,i,j);
            if(pop[i][j]->isMale)
46  count++;
        }
        if(count == args->p)
            pop[i][0]->isMale = 0;
        else if(count == 0)
```

```

51     pop[i][0]->isMale = 1;
    }
    for(i=0;i<args->p;i++)
        pop[args->g-1][i]->descendants[i] = 1;
    /* generate genealogy */
56    for(i=args->g-1;i>0;i--){
        for(j=0;j<args->p;j++){
            /* let descendants pick ancestors */
            pick1 = gsl_rng_uniform(ranf)*args->p;
            pick2 = gsl_rng_uniform(ranf)*args->p;
61    /*     while((pick2=gsl_rng_uniform(ranf)*args->p) == pick1) */
        /*     ; */
            while(pop[i-1][pick2]->isMale == pop[i-1][pick1]->isMale)
                pick2 = gsl_rng_uniform(ranf)*args->p;
            pop[i][j]->ancestor1 = pop[i-1][pick1];
66    pop[i][j]->ancestor2 = pop[i-1][pick2];
            if(gsl_rng_uniform(ranf) < 0.5)
                pop[i][j]->gene1->ancestor = pop[i-1][pick1]->gene1;
            else
                pop[i][j]->gene1->ancestor = pop[i-1][pick1]->gene2;
71    if(gsl_rng_uniform(ranf) < 0.5)
                pop[i][j]->gene2->ancestor = pop[i-1][pick2]->gene1;
            else
                pop[i][j]->gene2->ancestor = pop[i-1][pick2]->gene2;
            /* inherit descendants */
76    for(k=0;k<args->p;k++){
                if(pop[i][j]->descendants[k]){
                    pop[i-1][pick1]->descendants[k] = 1;
                    pop[i-1][pick2]->descendants[k] = 1;
                }
81    }
        }
    }
    /* check for universal ancestors & universal non-ancestors*/
    for(i=0;i<args->g;i++){
86    for(j=0;j<args->p;j++){
        count = 0;
        for(k=0;k<args->p;k++){
            if(pop[i][j]->descendants[k]){
                count++;
91    }
        }
        if(count == args->p){
            pop[i][j]->isUa = 1;
        }else if(count == 0){
96    pop[i][j]->isNonUa = 1;
        }
    }
}
/* compute tmrca */
101 k = 0;
    for(i=args->g-1;i>0;i--){
        for(j=0;j<args->p;j++){
            if(pop[i][j]->isUa){

```

```

106         pop[i][j]->isMrca = 1;
            k = 1;
        }
        if(k)
            break;
    }
111    tmrca = args->g - 1 - i;
    /* compute tia */
    for(i=args->g-1;i>0;i--){
        full = empty = 0;
        for(j=0;j<args->p;j++){
116            if(pop[i][j]->isUa)
                full++;
            else if(pop[i][j]->isNonUa)
                empty++;
            if(full > 0 && full+empty == args->p)
121                break;
        }
        tia = args->g - 1 - i;
        if(args->c){
            printf("Generations_to_the_most_recent_common_ancestor:_%d\n",tmrca);
126            printf("Generations_to_identical_ancestors:_____d\n",tia);
        }else{
            draw(args,pop);
            if(args->t == NULL)
                printf("To_view_the_graphic_in_file_%s,_include_it_in_a_LaTeX_file.\n
                    ",args->o);
131            else{
                fp = fopen(args->t,"w");
                printLatexHeader(fp);
                fprintf(fp,"\\begin{center}\\resizebox{\\textwidth}{!}{\\input{");
                i = 0;
136                j = strlen(args->o);
                while(args->o[i] != '.' && i < j)
                    fprintf(fp,"%c",args->o[i++]);
                fprintf(fp,"}\\end{center}\\n\\n\\end{document}");
                printf("To_view_the_graphic,_run_latex_%s\n",args->t);
141                fclose(fp);
            }
        }
        free_gsl_rng(ranf, args);
        return 0;
146    }

```

## 4 Change Log

- Version 1.3 (February 15, 2017)
  - Adjusted calling of random numbers.
  - Included documentation.
  - Adjusted the interface.
- Version 1.4 (November 6, 2018)

- Fixed bug in `interface.c`.