`numAl`, v. 0.5: DESCRIPTION

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

March 22, 2018

1 Introduction

2 Getting Started

`numAl` 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 numAl_XXX.tgz
  ```

  where `XXX` indicates the version.

- Change into the newly created directory

  ```
  cd NumAl_XXX
  ```

  and list its contents

  ```
  ls
  ```

- Generate `numAl`

  ```
  make
  ```

- List its options

  ```
  ./numAl -h
  ```

3 Listing

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

```
/***** numAl.c ***********************************
 * Description:
 * Author: Bernhard Haubold, haubold@evolbio.mpg.de
 * Date: Fri Sep 23 11:50:21 2016
 **************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
```

```c
#include "interface.h"
#include "eprintf.h"

double numAl(int l1, int l2, int printMatrix){
  int min,max;
  double **mat;
  int i, j = 0;

  if(l1<l2){
    min = l1;
    max = l2;
  }else{
    min = l2;
    max = l1;
  }

  mat = (double **)emalloc(2*sizeof(double *));
  mat[0] = (double *)emalloc((min+1)*sizeof(double));
  mat[1] = (double *)emalloc((min+1)*sizeof(double));

  for(i=0;i<=min;i++)
    mat[0][i] = 1;
  mat[1][0] = 1;

  if(printMatrix){
      for(i=0;i<=min;i++)
          printf("%8d",1);
      printf("\n");
  }

  for(i=0;i<max;i++){
    for(j=1;j<=min;j++){
      if(i % 2)
        mat[0][j] = mat[0][j-1] + mat[1][j-1] + mat[1][j];
      else
        mat[1][j] = mat[1][j-1] + mat[0][j-1] + mat[0][j];
    }
    if(printMatrix){
        for(j=0;j<=min;j++)
            if(i % 2)
                printf("%8d",(int)mat[0][j]);
            else
                printf("%8d",(int)mat[1][j]);
        printf("\n");
    }
  }
  if((i-1) % 2)
    return mat[0][min];
  else
    return mat[1][min];
}

double numAlRecursive(int l1, int l2){
  double a;
  if(l1>0 && l2 >0)
    a = numAlRecursive(l1-1,l2)+numAlRecursive(l1-1,l2-1)+numAlRecursive(l1
      ,l2-1);
  else
    a=1;
  return a;
}

int main(int argc, char *argv[]){
  char *version;
  double a;
  Args *args;
  long t;

  version = "0.4";
  setprogname2("numAl");
  args = getArgs(argc, argv);

  if(args->h){
    printUsage(version);
    exit(0);
  }
  if(args->v){
    printSplash(version);
    exit(0);
  }
  if(args->e){
    printUsage(version);
    exit(-1);
  }

  t = clock();
  if(args->t)
    a = numAlRecursive(args->m,args->n);
  else
    a = numAl(args->m,args->n,args->p);
  printf("f(%d,%d)␣=␣%.4e␣time␣=␣%.2f␣s\n",args->m,args->n,a,(double)(clock
     ()-t)/(double)CLOCKS_PER_SEC);

  free(args);
  free(progname());

  return 0;
}
```

4 Change Log

- Version 0.5 (September 23, 2016)
  - First version with modern argument processing. Version 0.4 did not work under OSX.
- Version 0.6 (March 22, 2018)
  - Put the `-t` option in square brackets.